

NASA CR-165,976

NASA Contractor Report 165976

NASA-CR-165976
19820026254

Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation

Appendix A. Software Documentation

J. W. Lowrie, Dr. A. J. Fermelia, D. C. Haley,
K. D. Gremban, J. Van Baalen, and R. W. Walsh

Martin Marietta Aerospace
Denver Aerospace
P.O. Box 179
Denver, Colorado 80201

Contract NAS1-16759
September 1982



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

RECEIVED
SEP 11 1982

LANGLEY RESEARCH CENTER

LANGLEY RESEARCH CENTER
HAMPTON, VIRGINIA



NF01885

NASA CR-165976

September 1982

Appendix A -
Software
Documentation

**EVALUATION OF AUTOMATED
DECISIONMAKING METHODOLOGIES
AND DEVELOPMENT OF AN
INTEGRATED ROBOTIC SYSTEM
SIMULATION**

Prepared by:

James W. Lowrie
Dr. Alfred J. Fermelia
Dennis C. Haley
Keith D. Gremban
Jeff Van Baalen
Richard W. Walsh

This work was performed for NASA
Langley Research Center under
contract NAS1-16759.

**MARTIN MARIETTA AEROSPACE
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201**

N82-34130#

FOREWORD

This document covers the work performed on contract NAS1-16759, Evaluation of Automated Decision-Making Methodologies and Development of Integrated Robotic System Simulation, for the Langley Research Center of the National Aeronautics and Space Administration. It was prepared by Martin Marietta Aerospace in accordance with the contract, Part II, Statement of Work.

The final report for this study consists of three volumes:

NASA CR-165975 - Study Results

NASA CR-165976 - Appendix A, Software Documentation

NASA CR-165977 - Appendix B, Derivation of Requirements Tool Dynamics
Appendix C, Derivation of Simulation Tool Dynamics
Appendix D, Derivation of Requirements Tool Control Law
Appendix E, Simulation Methodologies

Comments or requests for additional information should be directed to:

Jack Pennington
Mail No. 152D
Contracting Officer Representative
Langley Research Center
Hampton, VA 23665

or

James W. Lowrie
Mail No. 0570
Martin Marietta Aerospace
P.O. Box 179
Denver, CO 80201

CONTENTS

		Page
1.0	INTRODUCTION	A1-1
2.0	ROBSIM - EXECUTIVE PROGRAM	A2-1
2.1	System Definition Function Driver (INITDRVR)	A2-2
2.1.1	CREATE	A2-4
2.1.1.1	BASE	A2-6
2.1.1.2	JOINT	A2-8
2.1.1.3	LINK	A2-10
2.1.1.4	TOOL	A2-12
2.1.1.5	OBJECT	A2-13
2.1.1.6	GRAPH	A2-15
2.1.1.7	CNTROL	A2-17
2.1.1.8	PRGOPT	A2-19
2.1.2	BLDDAT	A2-21
2.1.2.1	CYL	A2-23
2.1.2.2	RECT	A2-26
2.1.2.3	TRISTR	A2-29
2.1.2.4	ORIENT	A2-31
2.1.2.4.1	MAT	A2-32
2.1.2.4.2	MATVEC	A2-33
2.1.2.5	DRAW	A2-34
2.1.2.5.1	ESMAT	A2-35
2.1.2.6	DBASE	A2-37
2.2	Analysis Tools Function Driver (SIMDRVR)	A2-38
2.2.1	REQUIR	A2-39
2.2.1.1	LDCOM	A2-41
2.2.1.2	GRAFIX, MATINT	A2-42
2.2.1.3	SEGMNT	A2-46
2.2.1.4	CNTRLR	A2-47
2.2.1.4.1	SETUP	A2-49
2.2.1.4.2	POUTC	A2-50
2.2.1.4.3	WROUT	A2-51
2.2.1.4.4	DYNAMICS	A2-53
2.2.1.4.4.1	TQBASE	A2-54
2.2.1.4.4.2	CABSM	A2-55
2.2.1.4.4.3	FORCE	A2-56
2.2.1.4.4.4	TORQUE	A2-57
2.2.1.4.4.5	DYNOUT	A2-58
2.3	Postprocessing Function Driver (POSTDRVR)	A2-60
2.3.1	MOTION	A2-61
2.3.2	GENPLT	A2-62
2.3.2.1	RDPLT	A2-65
2.3.2.2	LOGO	A2-66
2.4	ROBSIM Utilities	A2-67
2.4.1	ERRMSG	A2-68
2.4.2	RDSIM	A2-69
2.4.3	SETLU	A2-70
2.4.4	WRTSIM	A2-71
2.4.5	ZERCOM	A2-72
2.5	Math Utilities	A2-73

CONTENTS (cont)

		<u>Page</u>
2.5.1	CCTM	A2-73
2.5.2	SLVLIN, GAUSS	A2-74
2.5.3	MATMPY	A2-76
2.5.4	SKEW	A2-77
2.5.5	CETM	A2-78
2.6	Evans and Sutherland Graphics Routines	A2-79
2.7	DISSPLA Plot Routines	A2-81
3.0	JNTMOD - MAIN PROGRAM	A3-1
3.1	General Joint Model Routines	A3-5
3.1.1	INPUT	A3-5
3.1.2	INIT, PTINIT	A3-7
3.1.3	UVEC, COMPD	A3-9
3.1.4	XNOISE	A3-11
3.1.5	COMPC	A3-12
3.1.6	SENSOR	A3-13
3.1.6.1	HCOMP	A3-14
3.1.6.2	ZNOISE	A3-15
3.1.7	PRT	A3-16
3.2	Controller Model (CONTRL)	A3-17
3.2.1	COMPM	A3-19
3.2.2	LOADR	A3-20
3.2.3	PHIC	A3-21
3.2.4	THETAC	A3-22
3.2.5	CONTC	A3-23
3.2.6	CNOISE	A3-24
3.3	Amplifier Model (AMP)	A3-25
3.3.1	PHIA	A3-27
3.3.2	THETAA	A3-28
3.4	Motor Model (MOTOR)	A3-29
3.4.1	PHIM	A3-31
3.4.2	THETAM	A3-32
3.5	Power Train Model (PWRTRN)	A3-33
3.5.1	PHIP	A3-35
3.5.2	THETAP	A3-36
3.6	Load Model (LOAD)	A3-37
3.6.1	PHIL	A3-39
3.6.2	PHETAL	A3-40
3.7	Kalman Filter	A3-41
3.7.1	KFINIT	A3-42
3.7.2	BLDZAL	A3-43
3.7.3	BLDPHI	A3-44
3.7.4	BLDTHT	A3-45
3.7.5	BUILDH	A3-46
3.7.6	KALMAN	A3-47
3.7.6.1	BUILDG	A3-48
3.7.6.2	PRTKF	A3-49
3.8	Plot File Routines	A3-50
3.8.1	PLTSET	A3-51
3.8.2	LDVALU	A3-52

CONTENTS (cont)

		<u>Page</u>
3.8.3	WRTHDR	A3-53
3.9	Matrix Math Routines	A3-54
3.9.1	MATADD	A3-55
3.9.2	MATMPY	A3-56
3.9.3	MATTRN	A3-57
3.9.4	MATSUB	A3-58
3.10	IMSL Math Package Routines	A3-59

Figure

A1-1	ROBSIM Framework	A1-2
A1-2	Top-Level ROBSIM Hierarchy Diagram	A1-3
A1-3	System Definition Function Hierarchy Diagram	A1-4
A1-4	Analysis Tools Function Hierarchy Diagram	A1-5
A1-5	Postprocessing Function Hierarchy Diagram	A1-6
A1-6	JNTMOD Program Hierarchy Diagram	A1-7
A1-7	Initialization Hierarchy Diagram	A1-9
A1-8	Kalman Filter Initialization Hierarchy Diagram	A1-10
A1-9	Control Model Hierarchy Diagram	A1-10
A1-10	Amplifier Model Hierarchy Diagram	A1-11
A1-11	Motor Model Hierarchy Diagram	A1-12
A1-12	Power Train Model Hierarchy Diagram	A1-13
A1-13	Load Model Hierarchy Diagram	A1-14
A2-1	VCLR for ROBSIM Executive Program	A2-1
A2-2	INITDRVR VCLR	A2-3
A2-3	CREATE VCLR	A2-5
A2-4	BASE VCLR	A2-7
A2-5	JOINT VCLR	A2-9
A2-6	LINK VCLR	A2-11
A2-7	TOOL VCLR	A2-12
A2-8	OBJECT VCLR	A2-14
A2-9	GRAPH VCLR	A2-16
A2-10	CNTROL VCLR	A2-18
A2-11	PRGOPT VCLR	A2-20
A2-12	BLDDAT VCLR	A2-22
A2-13	Cylinder and Cone Shape Coordinate Systems	A2-24
A2-14	CYL VCLR	A2-25
A2-15	Rectangular and Trapezoidal Shape Coordinate Systems	A2-27
A2-16	RECT VCLR	A2-28
A2-17	Triangular Cross-Section Beam Coordinate System	A2-29
A2-18	TRISTR VCLR	A2-30
A2-19	ORIENT VCLR	A2-31
A2-20	MAT VCLR	A2-32
A2-21	MATVEC VCLR	A2-33
A2-22	DRAW VCLR	A2-34
A2-23	ESMAT VCLR	A2-36
A2-24	DBASE VCLR	A2-37
A2-25	SIMDRVR VCLR	A2-38
A2-26	REQUIR VCLR	A2-40
A2-27	LDCOM VCLR	A2-41

Figure (cont)

		<u>Page</u>
A2-28	GRAFIX VCLR	A2-43
A2-29	MATINT VCLR	A2-45
A2-30	SEGMNT VCLR	A2-46
A2-31	CNTRLR VCLR	A2-48
A2-32	SETUP VCLR	A2-49
A2-33	POUTC VCLR	A2-50
A2-34	WROUT VCLR	A2-52
A2-35	DYNAMICS VCLR	A2-53
A2-36	TQBASE VCLR	A2-54
A2-37	CABSM VCLR	A2-55
A2-38	FORCE VCLR	A2-56
A2-39	TORQUE VCLR	A2-57
A2-40	DYNOUT VCLR	A2-59
A2-41	POSTDRV VCLR	A2-60
A2-42	MOTION VCLR	A2-61
A2-43	GENPLT VCLR	A2-63
A2-44	RDPLT VCLR	A2-65
A2-45	LOGO VCLR	A2-66
A2-46	ERRMSG VCLR	A2-68
A2-47	RDSIM VCLR	A2-69
A2-48	SETLU VCLR	A2-70
A2-49	WRTSIM VCLR	A2-71
A2-50	ZERCOM VCLR	A2-72
A2-51	CCTM VCLR	A2-73
A2-52	SLVLIN VCLR	A2-74
A2-53	GAUSS VCLR	A2-75
A2-54	MATMPY VCLR	A2-76
A2-55	SKEW VCLR	A2-77
A2-56	CETM VCLR	A2-78
A3-1	Typical Model Block	A3-2
A3-2	Joint Model	A3-2
A3-3	JNTMOD VCLR	A3-4
A3-4	INPUT VCLR	A3-6
A3-5	INIT VCLR	A3-7
A3-6	PTINIT VCLR	A3-8
A3-7	UVEC VCLR	A3-9
A3-8	COMPD VCLR	A3-10
A3-9	XNOISE VCLR	A3-11
A3-10	COMPC VCLR	A3-12
A3-11	SENSOR VCLR	A3-13
A3-12	HCOMP VCLR	A3-14
A3-13	ZNOISE VCLR	A3-15
A3-14	PRT VCLR	A3-16
A3-15	CONTRL VCLR	A3-18
A3-16	COMPM VCLR	A3-19
A3-17	LOADR VCLR	A3-20
A3-18	PHIC VCLR	A3-21
A3-19	THETAC VCLR	A3-22
A3-20	CONTC VCLR	A3-23
A3-21	CNOISE VCLR	A3-24

Figure (cont)

		<u>Page</u>
A3-22	AMP VCLR	A3-26
A3-23	PHIA VCLR	A3-27
A3-24	THETAA VCLR	A3-28
A3-25	MOTOR VCLR	A3-30
A3-26	PHIM VCLR	A3-31
A3-27	THETAM VCLR	A3-32
A3-28	PWRTRN VCLR	A3-34
A3-29	PHIP VCLR	A3-35
A3-30	THETAP VCLR	A3-36
A3-31	LOAD VCLR	A3-38
A3-32	PHIL VCLR	A3-39
A3-33	THETAL VCLR	A3-40
A3-34	KFINIT VCLR	A3-42
A3-35	BLDZAL VCLR	A3-43
A3-36	BLDPHI VCLR	A3-44
A3-37	BLDTHT VCLR	A3-45
A3-38	BUILDH VCLR	A3-46
A3-39	KALMAN VCLR	A3-47
A3-40	BUILDG VCLR	A3-48
A3-41	PRTKF VCLR	A3-49
A3-42	PLTSET VCLR	A3-51
A3-43	LDVALU VCLR	A3-52
A3-44	WRTHDR VCLR	A3-53
A3-45	MATADD VCLR	A3-55
A3-46	MATMPY VCLR	A3-56
A3-47	MATTRN VCLR	A3-57
A3-48	MATSUB VCLR	A3-58

1.0 INTRODUCTION

The Robotic Simulation Program (ROBSIM) has been designed to provide a wide range of computer capabilities in robotic system design and analysis. The program structure is composed of three major functions controlled by a program executive as shown in Figure A1-1. The three major ROBSIM functions are:

- 1) System definition;
- 2) Analysis tools;
- 3) Postprocessing.

Each of the major functions is designed in a modular fashion to allow for easy future expansion.

The System Definition function handles user input of system parameters and creates a disk file to be used as input to the Analysis Tools and Postprocessing functions. The Analysis Tools function handles the computational requirements of the ROBSIM program. Currently, a requirements analysis tool and a simulation tool are being implemented. Additional analysis tools are easily included within the program structure. The simulation tool is being developed outside the ROBSIM program structure as a standalone program. The documentation of the simulation tool program is contained in a separate section of this document. The Postprocessing function allows for more detailed study of the results of the Analysis Tools function execution. Current postprocessing capabilities include a playback of the system motion using the program graphics and generation of parameter versus parameter plots.

1.1 DOCUMENT STRUCTURE

The remainder of Section 1.0 provides hierarchy diagrams for the software developed to date under the ROBSIM contract and a short discussion of the visual control logic representation (VCLR) diagrams used to show program logic flow.

Section 2.0 contains the documentation of the subroutines that make up the ROBSIM program and presents a description of the ROBSIM executive. Subsections 2.1 through 2.3 contain descriptions of the routines required in the System Definition, Analysis Tools, and Postprocessing functions, respectively; Subsection 2.4, descriptions of several program utility routines used throughout the ROBSIM program; Subsection 2.5, descriptions of several math utilities used throughout the ROBSIM program; Subsections 2.6 and 2.7, short descriptions of the Evans and Sutherland graphics routines and the DISSPLA plotting routines used in ROBSIM.

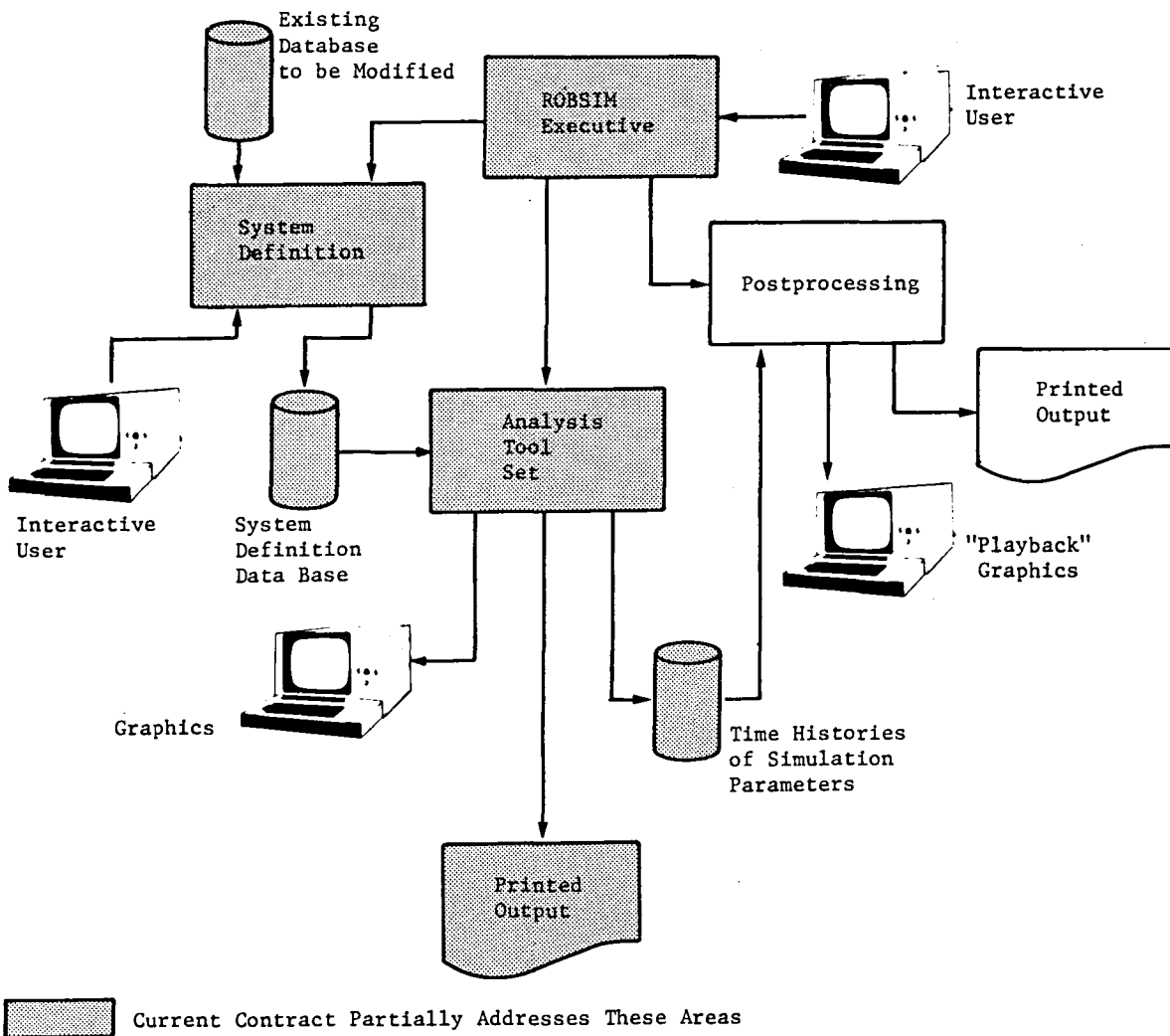


Figure A1-1 ROBSIM Framework

Section 3.0 contains the documentation of the subroutines that make up the simulation tool program, Joint Model (JNTMOD). Subsection 3.1 contains descriptions of the input routines, the initialization routine, the print routine, and several routines used in multiple locations in the JNTMOD program; Subsection 3.2, descriptions of the control routines; Subsections 3.3 through 3.6, descriptions of amplifier model, motor model, power train, and load model routines, respectively; Subsection 3.7, descriptions of the Kalman filter routines; Subsection 3.8, descriptions of the plot file generation routines; and Subsection 3.9, descriptions of the math routines.

The documentation of each routine in Sections 2.0 and 3.0 contains a general discussion of the function performed and a VCLR for that routine.

1.2 HIERARCHY DIAGRAMS

Hierarchy diagrams for the ROBSIM and JNTMOD programs are given in Figures A1-2 through A1-13. The numbers associated with each block in the hierarchy diagrams indicate the section number within this document containing the description of the subroutine that handles that block function. Figure A1-2 shows the top-level hierarchy diagram for the ROBSIM program. Figures A1-3 through A1-5 show the hierarchy diagrams for the three major ROBSIM functions--System Definition, Analysis Tools, and Postprocessing, respectively. Figures A1-6 through A1-13 show the hierarchy diagrams for the JNTMOD program.

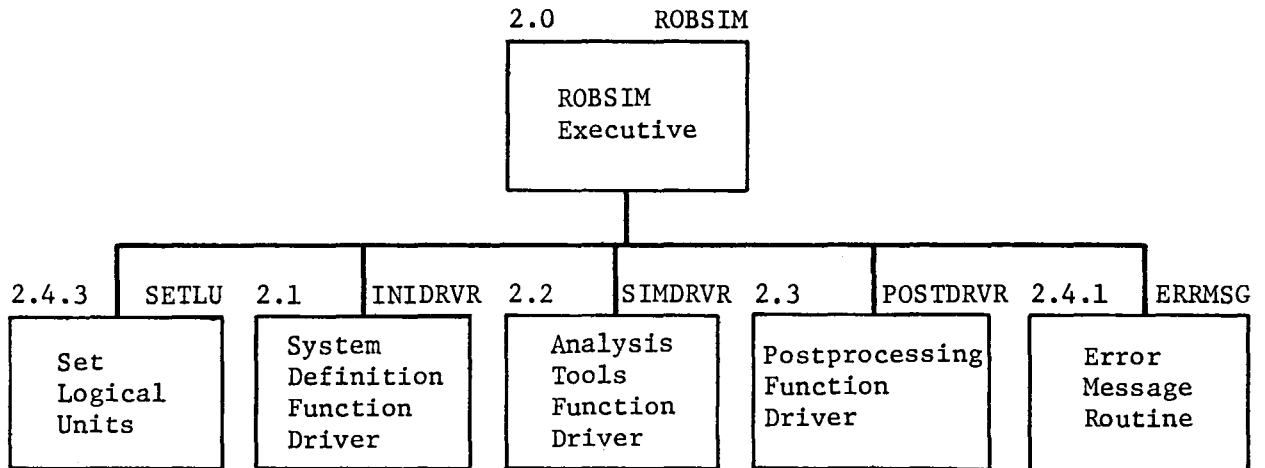
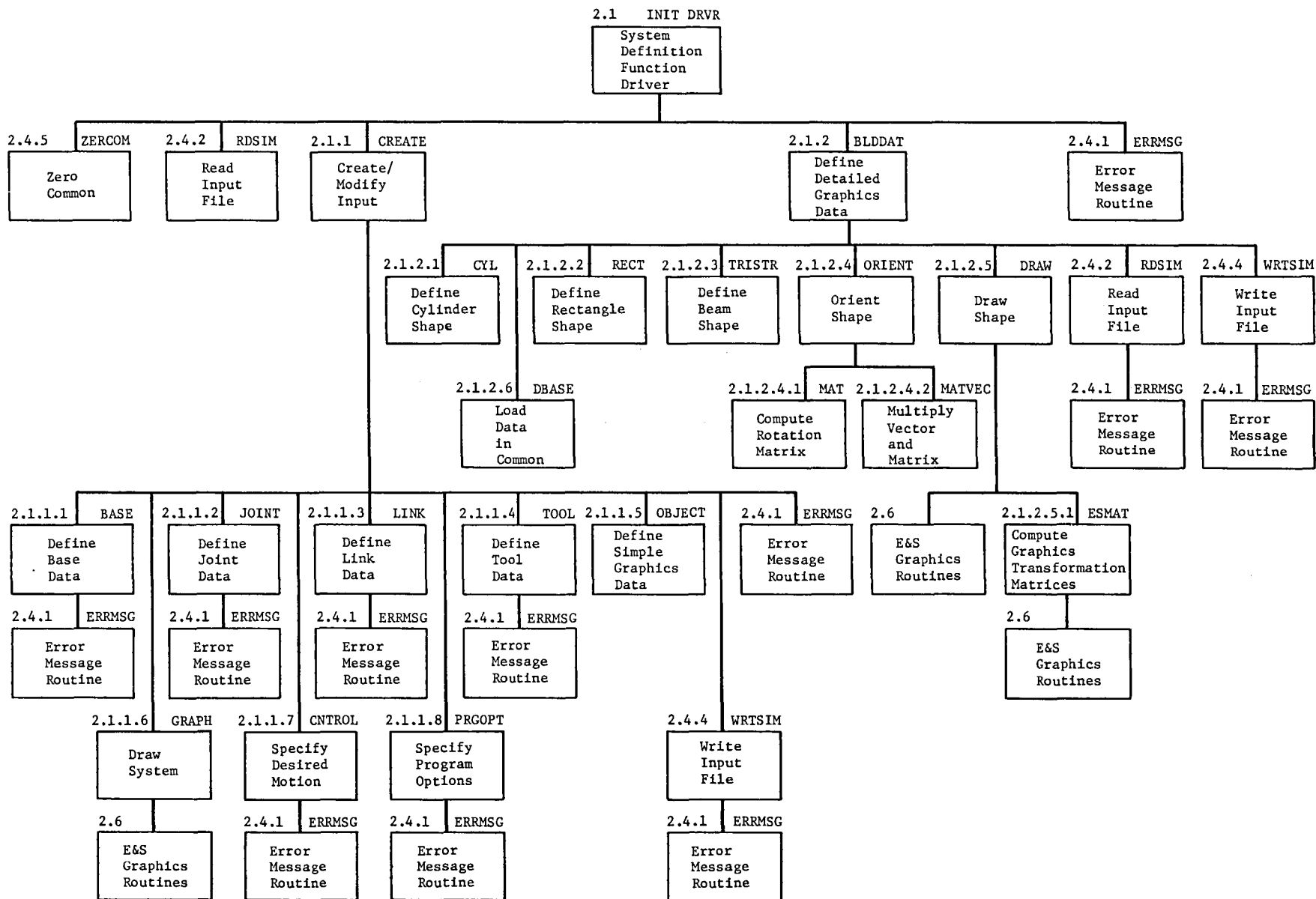


Figure A1-2 Top-Level ROBSIM Hierarchy Diagram

Figure A1-3



A1-4

Figure A1-3 System Definition Function Hierarchy Diagram

Figure AI-4

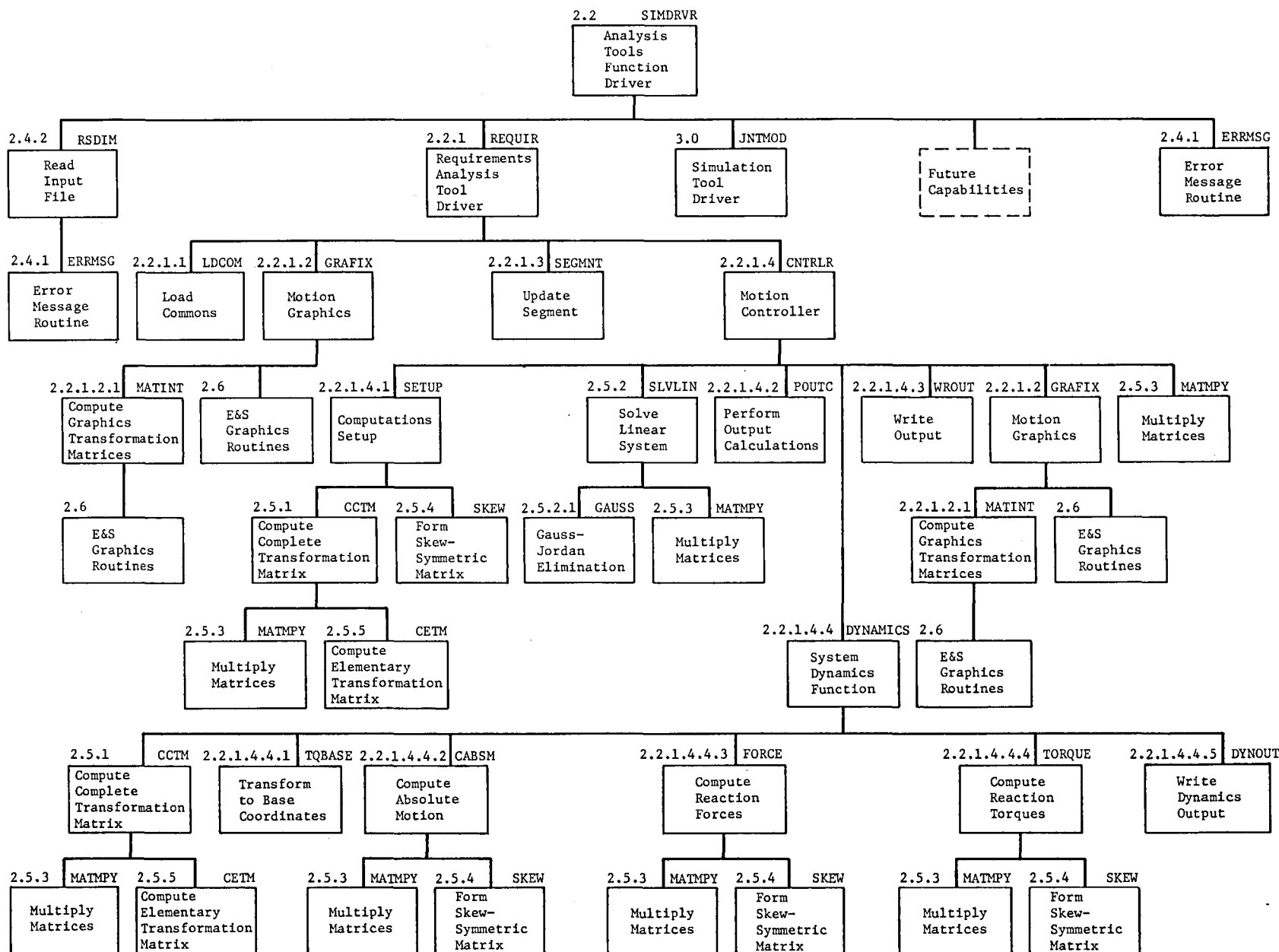


Figure AI-4

Figure AI-4 Analysis Tools Function Hierarchy Diagram

Figure A1-5

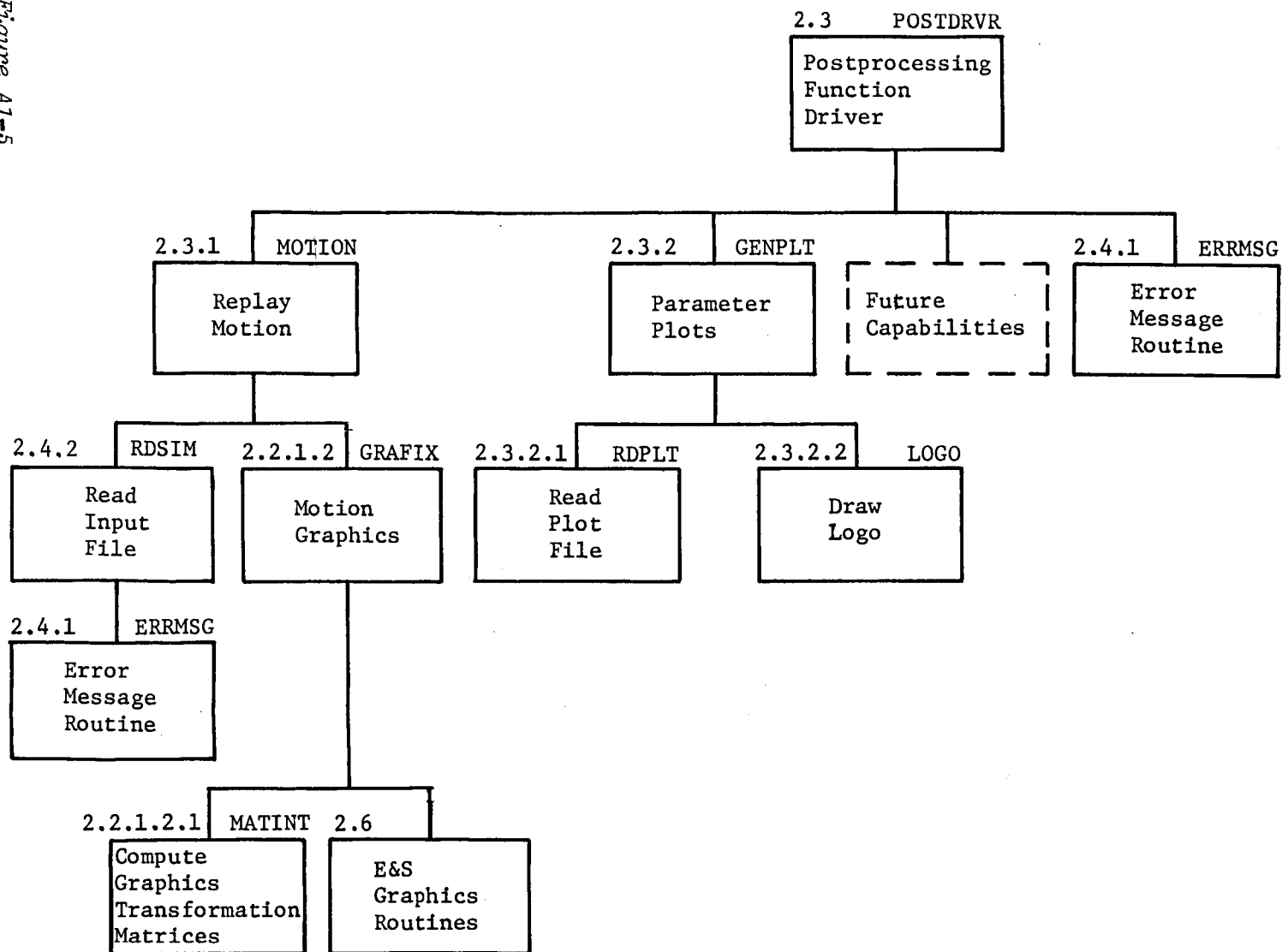


Figure A1-5 Postprocessing Function Hierarchy Diagram

Figure A1-6

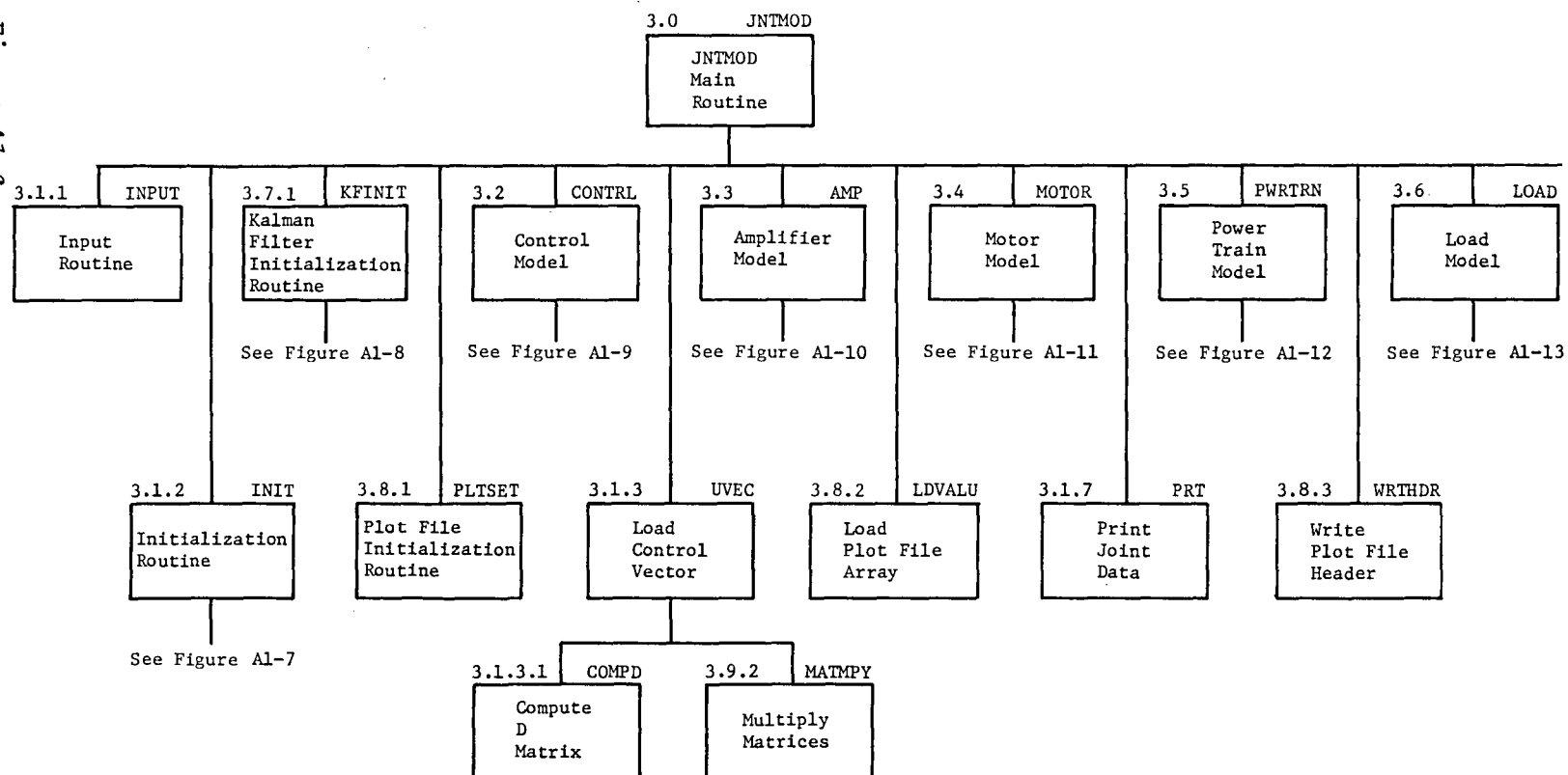


Figure A1-6 JNTMOD Program Hierarchy Diagram

Figure A1-6 (concl)

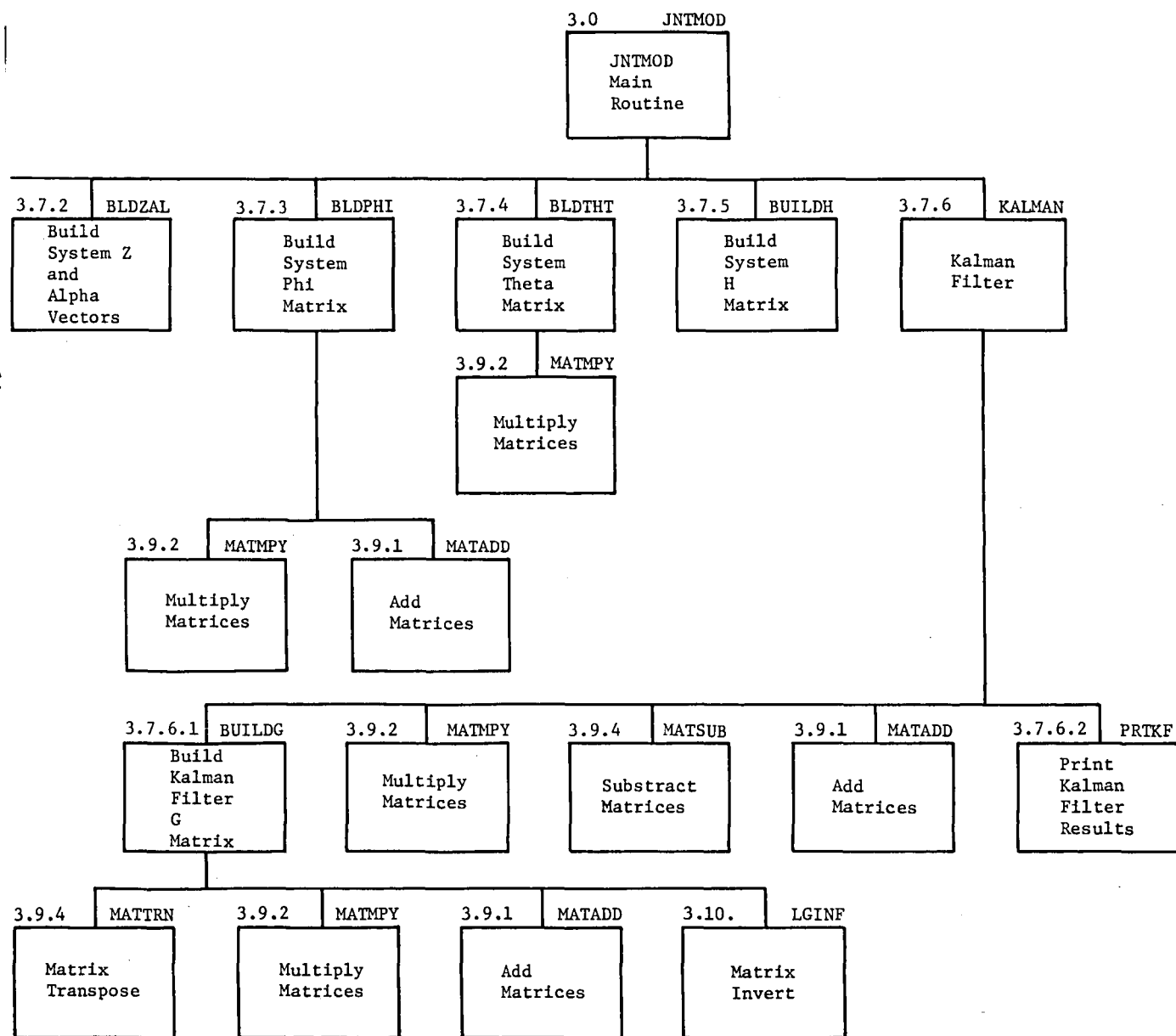


Figure A1-6 (concl)

Figure A1-7

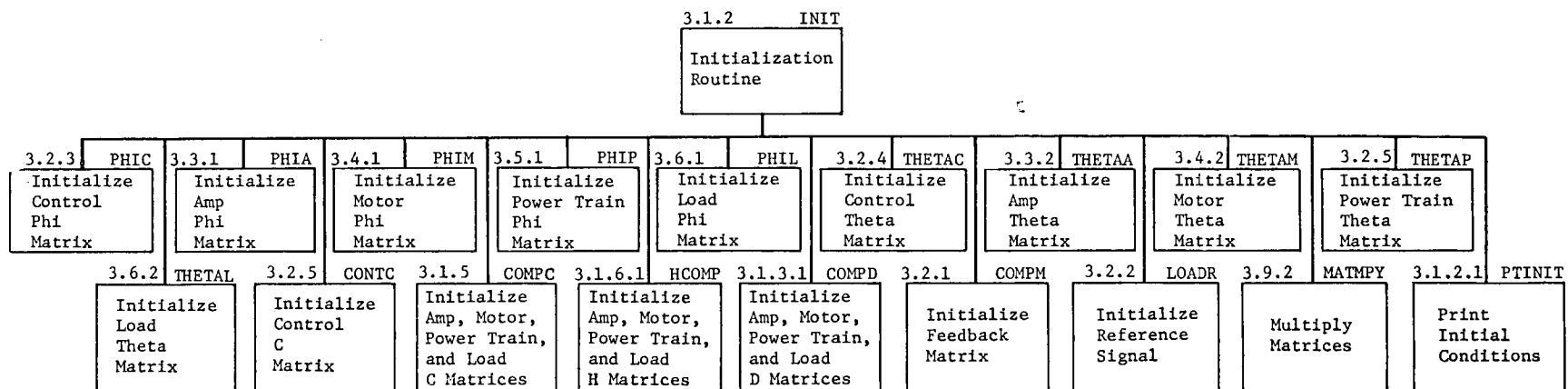


Figure A1-7 Initialization Hierarchy Diagram

Figure A1-7

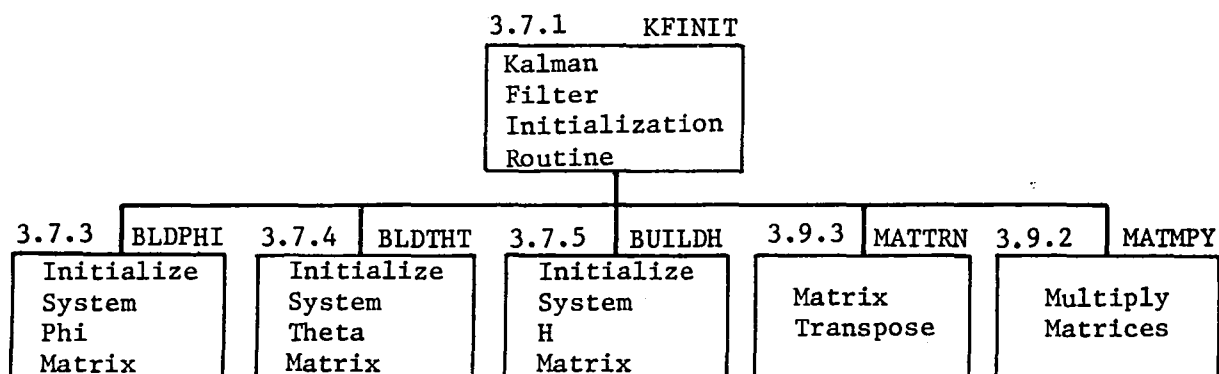


Figure A1-8 Kalman Filter Initialization Hierarchy Diagram

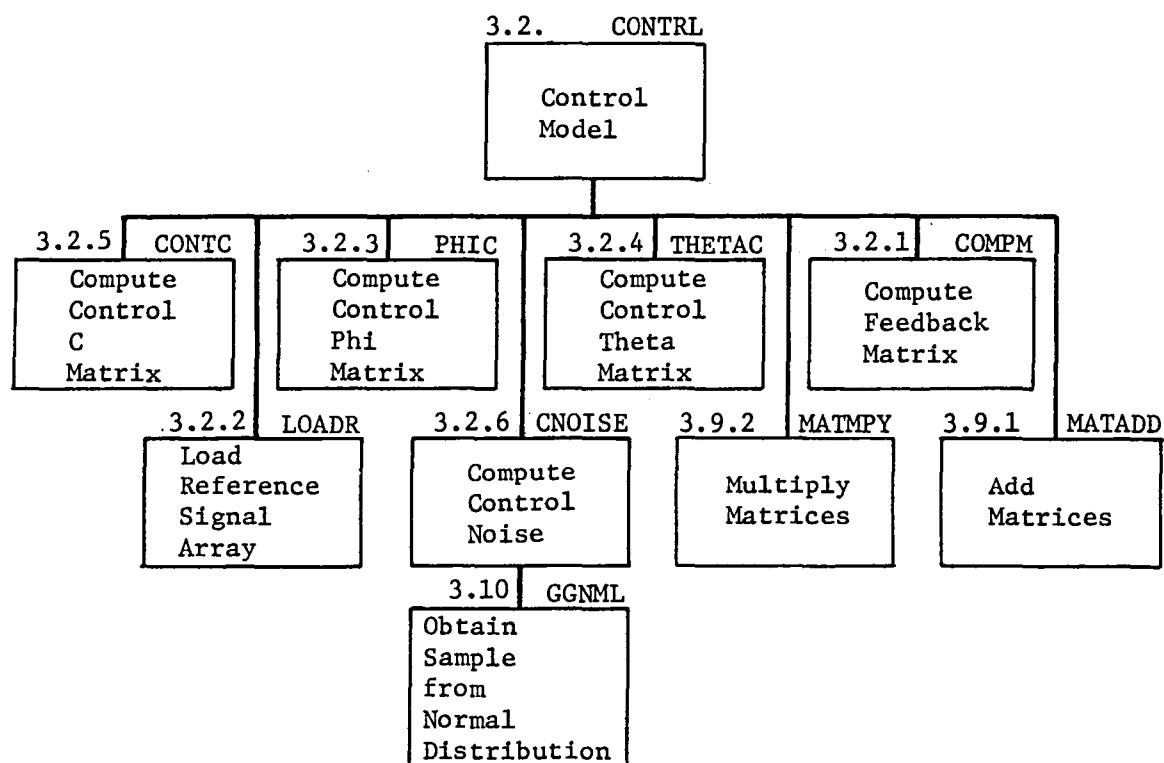


Figure A1-9 Control Model Hierarchy Diagram

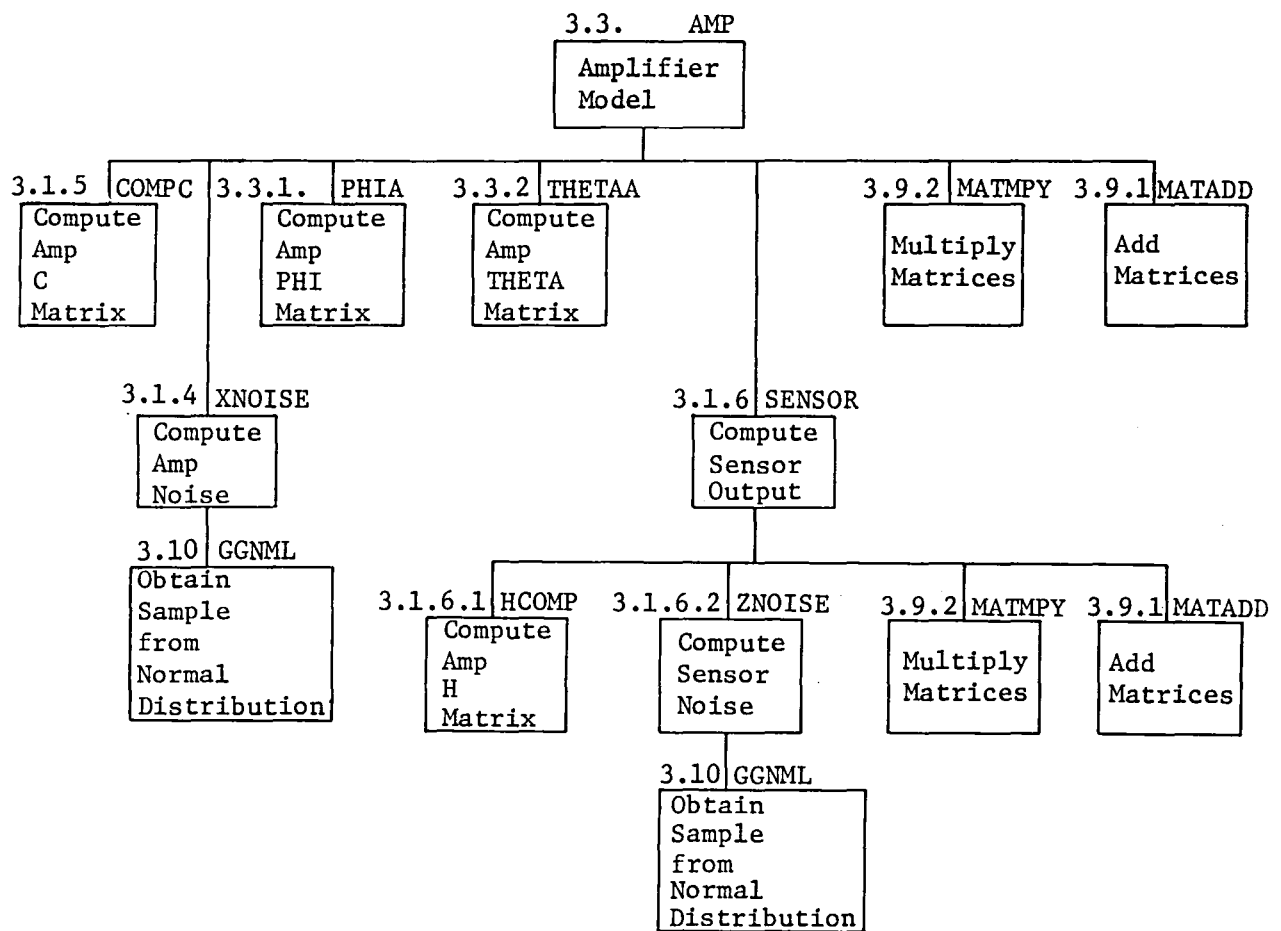


Figure A1-10 Amplifier Model Hierarchy Diagram

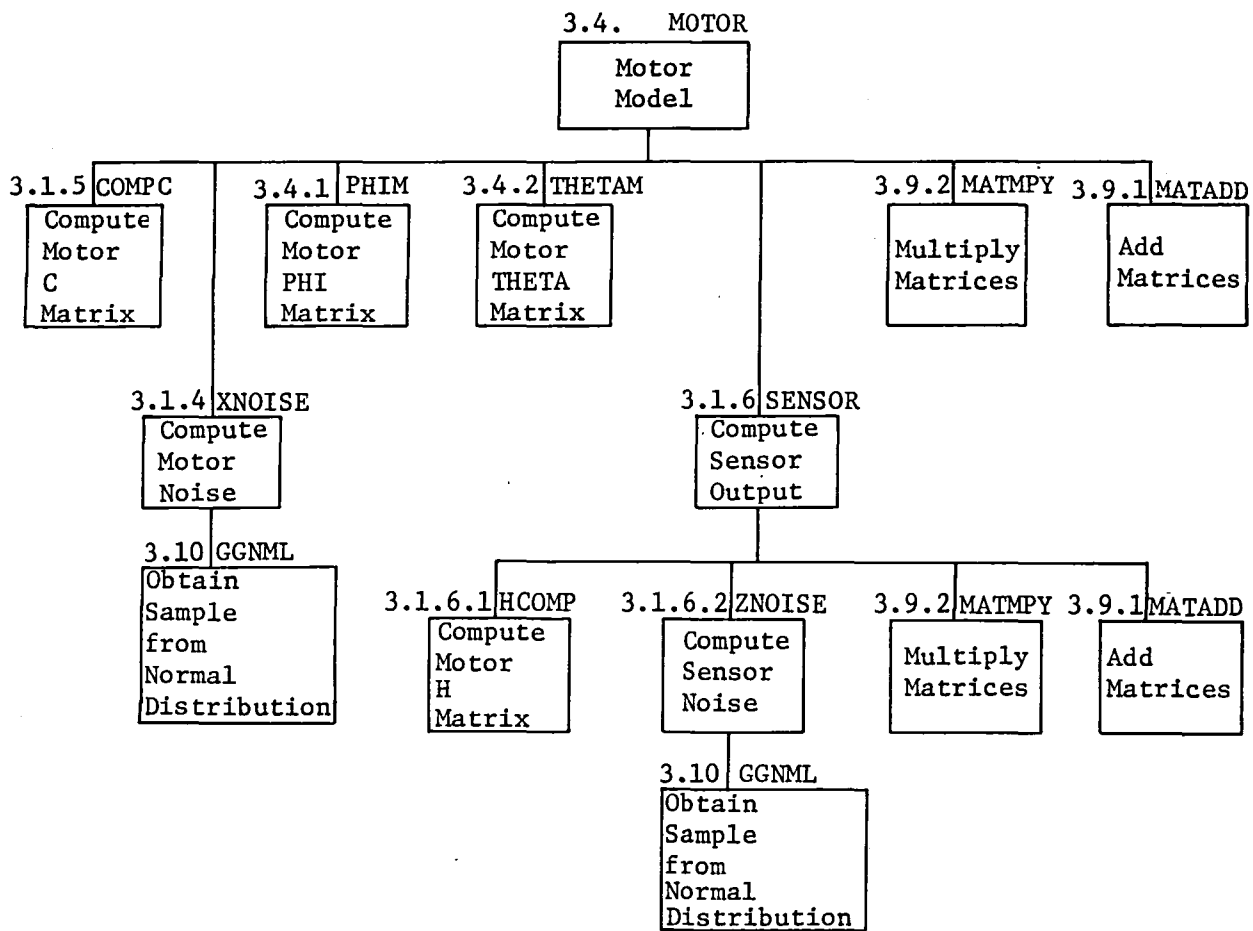


Figure A1-11 Motor Model Hierarchy Diagram

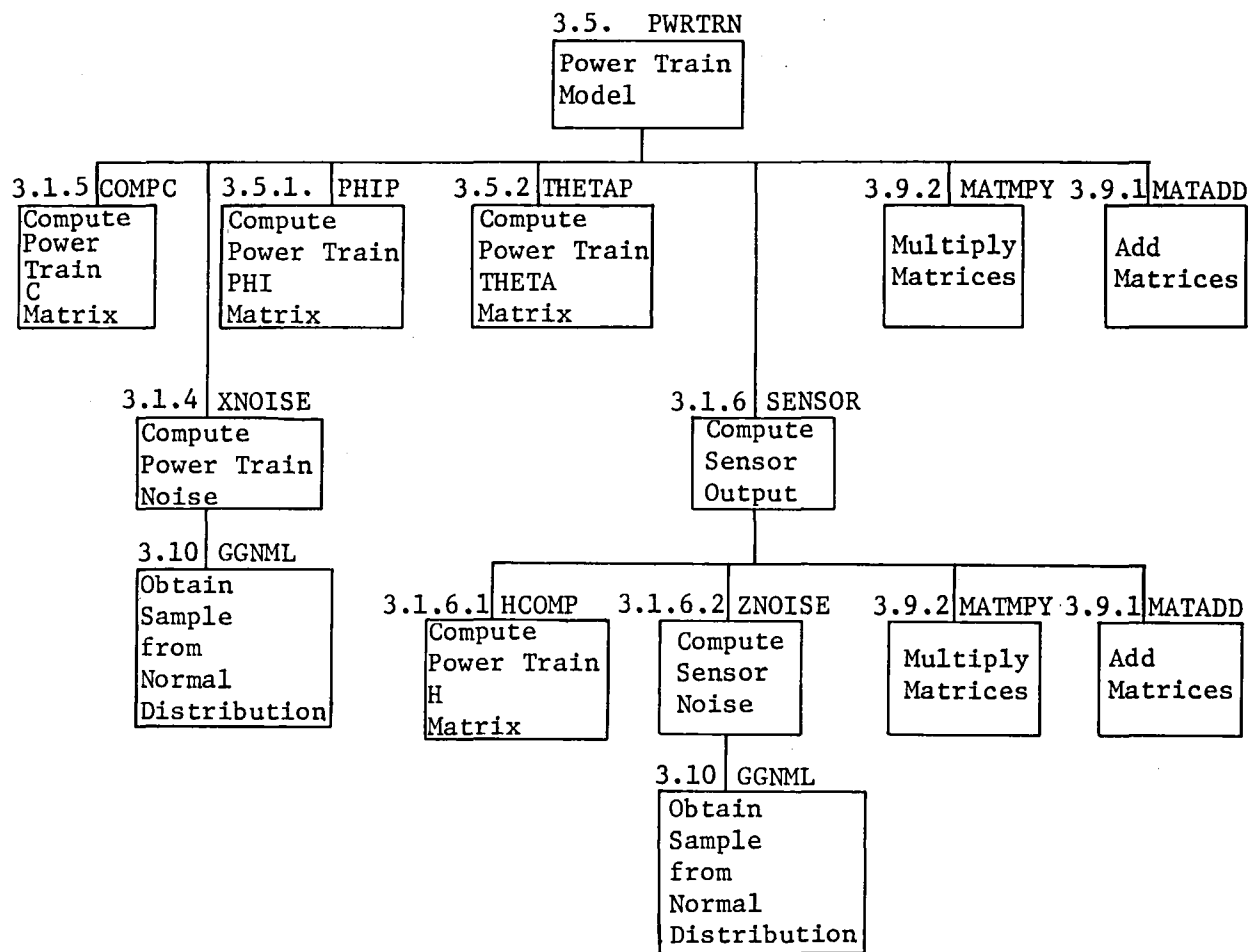


Figure A1-12 Power Train Model Hierarchy Diagram

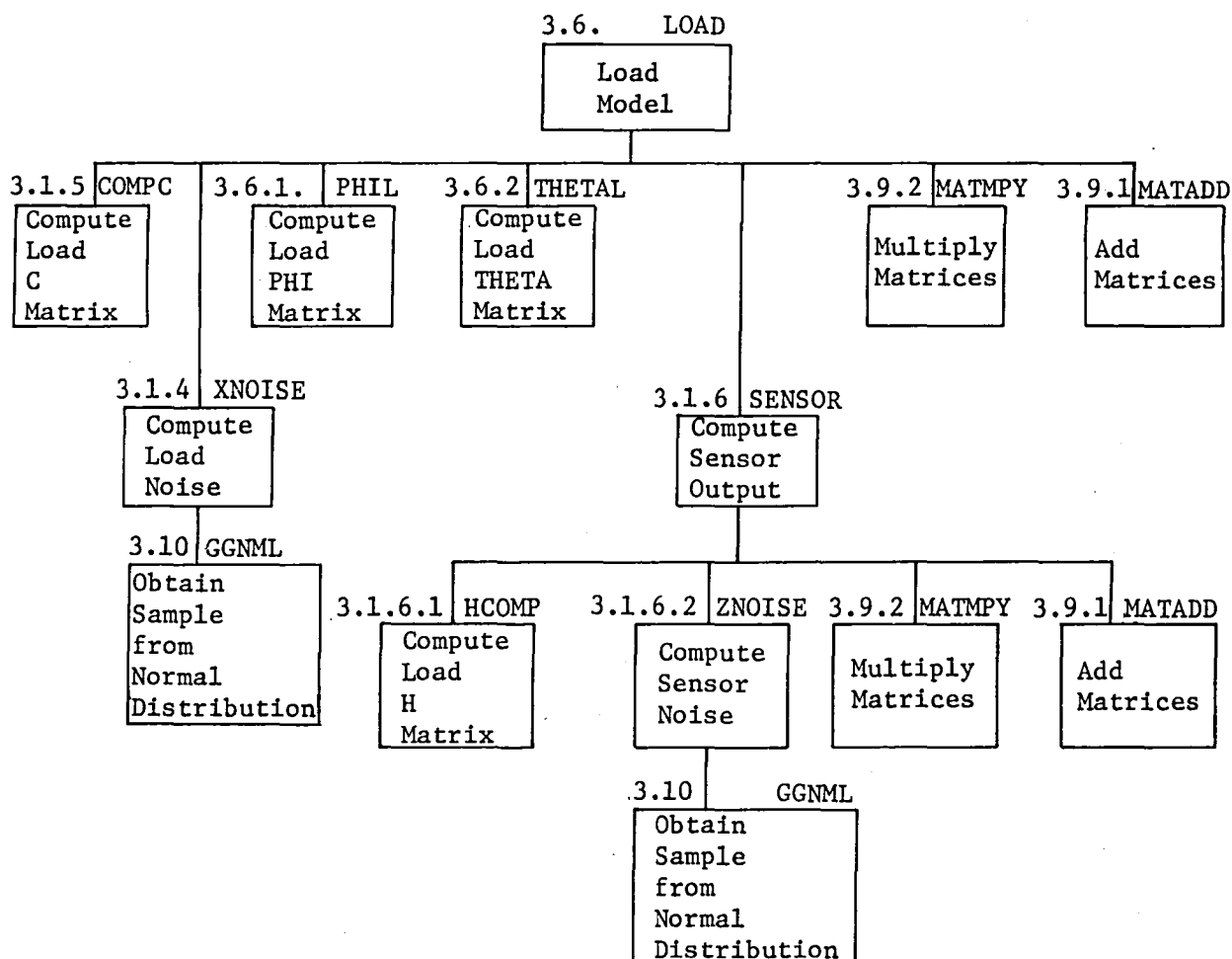


Figure A1-13 Load Model Hierarchy Diagram

1.3 VCLR FORMAT

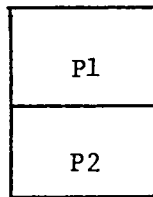
VCLR diagrams present program logic flow that is compatible with structured programming. The use of VCLR diagrams offers many advantages over the use of flow charts:

- 1) Only the standard constructs are used;
- 2) The total scope and impact of the logic can be seen and easily understood;
- 3) No extraneous symbols, connections, or notations are used.

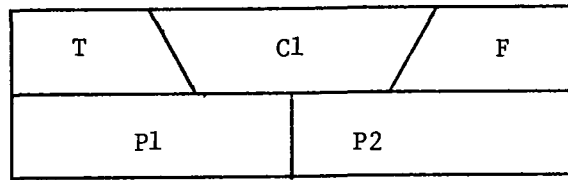
VCLR provides visible control logic representation, which is a picture of a software design. It enables software engineers to express their thinking visually and stresses the control logic of the design.

Standard constructs in visible control logic representations are the same as those for pseudo-code: SEQUENCE, IFTHENELSE, DOWHILE, DUNTIL, and DOCASE; only the representations differ.

SEQUENCE - A SEQUENCE is simply one standard construct or one single statement followed by another. If P1 and P2 are standard constructs or single statements, the sequence would appear in a visible control logic representation as:



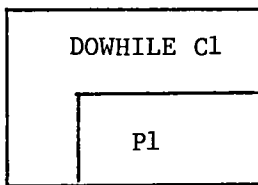
IFTHENELSE - IFTHENELSE consists of a true/false test and a path for each state. The true path appears on the left side, under the "T." One of the paths may be a "do nothing" or "NULL" path. One or both paths must consist of a standard construct or of a single statement. If "C1" is the condition being tested, "P1" is the true path, and "P2" is the false path, the IFTHENELSE construct would be written as:



DOWHILE - The DOWHILE is a loop with these characteristics:

- The counter or other item to be "incremented" is initialized before entering the loop.
- The test is performed at the beginning of the loop. The conditions that must exist for the loop to be executed are the conditions that appear in the DOWHILE test.
- The item to be executed must be a standard construct or a single statement.
- The counter is incremented or other increment-like action is generally taken (e.g., another line is read) at the end of the loop.

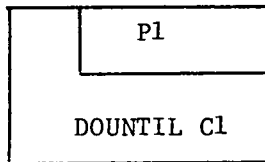
If "C1" is the condition that must exist for the loop to be executed, and "P1" is a standard construct or single statement, the DOWHILE would be written as follows:



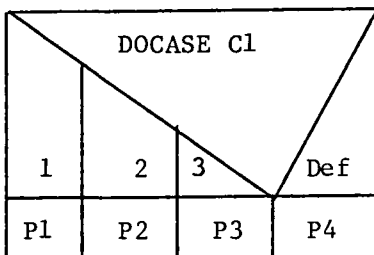
DOUNTIL - The DOUNTIL is a loop with these characteristics:

- The counter or other item to be "incremented" is initialized before entering the loop.
- The test is performed at the end of the loop. The conditions that must exist to exit from the loops are those that appear in the DOUNTIL test.
- The item to be executed must be a standard construct or a **single** statement.
- The counter is incremented or other increment-like action is generally taken (e.g., another line is read) at the beginning of the loop.

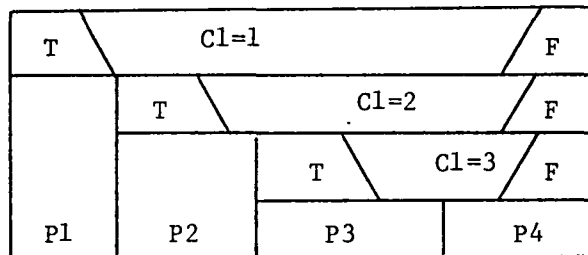
If "C1" is the condition that must exist to exit from the loop and "P1" is a standard construct or single statement, the DOUNTIL would be written as follows:



DOCASE - The DOCASE construct is for executing a different set of statements for each of several different values of a variable. If "C1" is the variable being tested and if "C1" may have values of 1, 2, or 3, the construct appears as follows:



Example A



Example B

Example A is equivalent to the nested IFTHENELSE form shown in B.

The executive routine controls the program execution through an interactive user prompt for the program function desired. Execution of the requested function is accomplished through a subroutine call to the appropriate function driver. The three program functions and the associated drivers are:

- 1) System definition (INITDRVR);
- 2) Analysis tools (SIMDRVR);
- 3) Postprocessing (POSTDRVR).

The System Definition function handles user input of system parameters and creates a disk file to be used as input to the Analysis Tools and Postprocessing functions. The Analysis Tools function handles the computational requirements of the ROBSIM program. The Postprocessing function allows for more detailed study of the results of the Analysis Tools function execution. Upon completion of the requested function execution, control is returned to the program executive. The program function prompt is then reissued to allow the user to either request another program function execution or request program termination.

Nonrecoverable errors encountered within any function return control to the executive program for display of the appropriate error message through a call to subroutine ERRMSG. Following a nonrecoverable error, the user may elect to terminate the program or reissue the program function prompt and attempt further program execution.

Figure A2-1 is the VCLR for the ROBSIM executive program.

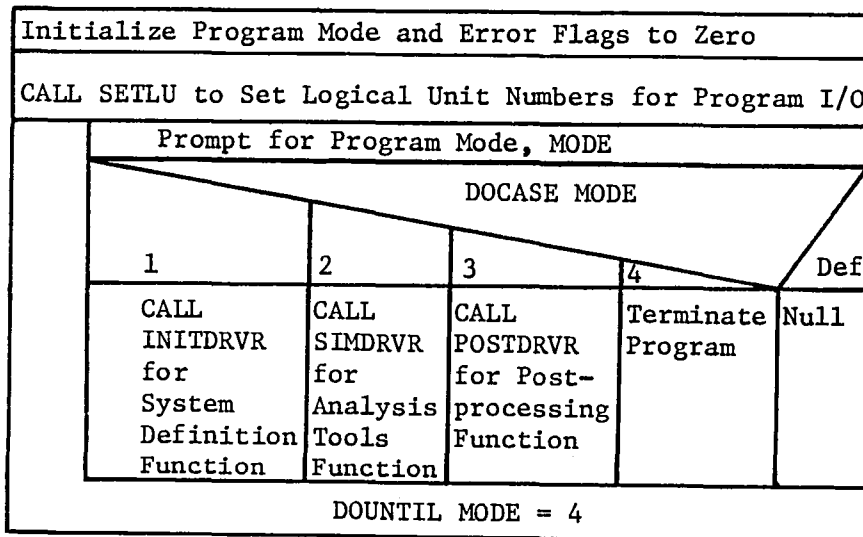


Figure A2-1 VCLR for ROBSIM Executive Program

2.1 SYSTEM DEFINITION FUNCTION DRIVER (INITDRVR)

The System Definition function driver operates in an interactive mode and prompts the user for the system definition option desired. Valid options are:

- 1) Create a new basic data file;
- 2) Modify an existing basic data file;
- 3) Specify detailed environment geometry;
- 4) Specify detailed system geometry;
- 5) Return to the ROBSIM executive.

Option 1 provides for the input of data describing a robotic system not previously studied. All data required to describe the system must be input through terminal responses to interactive prompts issued by various routines within the System Definition function. The result of executing Option 1 will be a disk file containing all data input describing the robotic system. No data describing the geometry of the environment are requested within the prompts of Option 1. The data used to describe the geometry of the robotic system for graphics display will consist of simple cylinder representations. The simple cylinder representation was chosen for ease of data point computation, ease of modification, and acceptability of use as a coarse representation of most robotic system components.

Option 2 provides for the modification of an existing data file previously created by the System Definition function. Through interactive prompts issued by various routines within the System Definition function, the user selects the data to be modified. As in Option 1, geometric data describing the environment are not input within Option 2. The use of simple cylinder representations for the robotic system components allows the addition and deletion of links for quick study of various system configurations. Option 2 may be selected for modification of an existing data file that contains detailed geometric data describing the robotic system. However, modifications should be limited to data not pertaining to the system geometry or configuration of joints and links. Modifications in these areas will destroy the detailed data and replace it with simple cylinder data.

Option 3 allows the user to describe the geometry of the environment for graphics display. The data describing the environment are built up of components that are simple three-dimensional shapes. A data file generated with Option 1 or 2 must exist prior to selection of Option 3.

Option 4 allows the user to specify a detailed geometric representation of the robotic system used in Option 1 and 2. A data file generated with Option 1 or 2 must exist prior to selecting Option 4. Option 4 should be used only when the system configuration is stable because

changes in the configuration require redefinition of the detailed geometry of the entire system.

If Option 1 is requested, subroutine ZERCOM is called to zero all COMMON locations used for data storage during execution of the System Definition function. Subroutine CREATE is then called to control the program flow and the creation of the data file.

If Option 2 is requested, subroutine RDSIM is called to read the existing data file and load the data into the appropriate COMMON blocks. Subroutine CREATE is then called to control the program flow and the modifications of data as required.

If either Option 3 or 4 is requested, subroutine BLDDAT is called to control the input of the detailed geometric data.

Recoverable errors encountered within subroutine INITDRVR cause an error message to be written through a call to subroutine ERRMSG followed by appropriate recovery action. Nonrecoverable errors encountered within routines called by INITDRVR cause return of control to the ROBSIM executive program.

Figure A2-2 is the VCLR for subroutine INITDRVR.

Prompt for System Definition Function Mode, IMODE					
DOCASE IMODE					
1	2	3	4	5	Def
CALL ZERCOM to Zero Common Locations	CALL RDSIM to Read Input File	CALL BLDDAT to Define Detailed Environment Graphics Data	CALL BLDDAT to Define Detailed Robotic System Graphics Data	Return to ROBSIM Executive	Null
CALL CREATE to Create New Input File	CALL CREATE to Modify Existing Input File				

Figure A2-2 INITDRVR VCLR

2.1.1 Create/Modify Data File (CREATE)

Subroutine CREATE controls the program flow during the creation or modification of the input data file under the System Definition function. Upon entering subroutine CREATE, the user turns on or off via interactive prompt the use of the graphics package. Subroutine CREATE is called in one of two modes, creation of a new data file, or modification of an existing file. In the create mode, the program logic flows sequentially through subroutine CREATE requiring input for all possible data. In the modify mode, the user is prompted to specify which data are to be changed. Program control is then sent directly to the routine responsible for those data.

The robotic system geometry and mass properties data are input through subroutines BASE, JOINT, LINK, and TOOL. The location orientation and size of the system base are defined in subroutine BASE. The location, orientation, size, and mass properties of each joint/link combination are defined in subroutines JOINT and LINK. Subroutine TOOL defines the location and orientation of the end effector. For the base, each link, and the end effector, subroutine OBJECT is called to generate the simple cylinder data for graphics representations. If graphics is requested, each system component is displayed as the data are input. In either the create or modify modes, the user may iterate on a particular component until it is correct before proceeding. Subroutine CNTROL allows the user to define a particular motion sequence for the robotic system. The program start time, stop time, time step, and other program option flags are set in subroutine PRGOPT. When all input is complete, subroutine WRTSIM is called to write the data file to disk.

Recoverable errors encountered within subroutine CREATE cause an error message to be written through a call to subroutine ERRMSG followed by appropriate recovery action. Nonrecoverable errors encountered within routines called by CREATE cause return of program control through the System Definition function routines back to the ROBSIM executive program.

Figure A2-3 is the VCLR for subroutine CREATE.

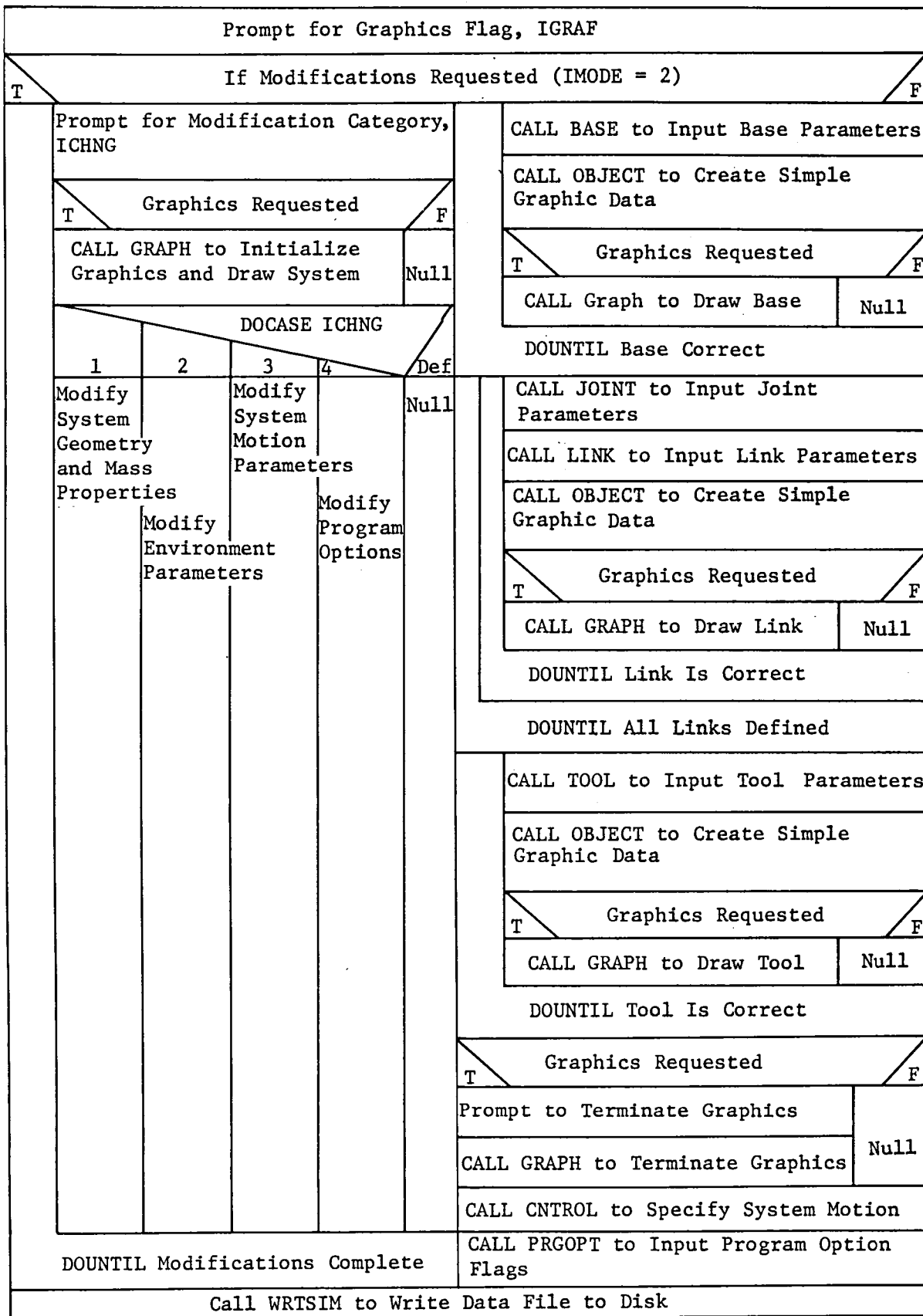


Figure A2-3 CREATE VCLR

2.1.1.1 Define Base Data (BASE) - Subroutine BASE interactively prompts the user for location, orientation, and size data for the robotic system base. The calling argument MOD specifies whether subroutine BASE was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated allowing modification of other base data. Upon completion of all changes, the user requests termination of modifications and control is returned to the calling program.

The location of the base is specified as the Cartesian coordinates of the origin of the base coordinate system given in terms of the "world" coordinate system. Using the graphics screen as a reference, the "world" coordinate system is defined with origin at the center of the screen, x-axis positive to the right, y-axis positive upward, and z-axis positive out of the screen.

The orientation of the base is specified as a rotation sequence and corresponding set of rotation angles. This rotation sequence and the associated rotation angles relate the base coordinate system axes to the "world" coordinate system axes. Performing the rotation sequence on the "world" system produces the base system. The base system should be oriented so that the x-axis points toward the first joint in the robotic system. Orientation of the y-z plane is arbitrary.

The size parameters the user is prompted for in subroutine BASE are used by subroutine OBJECT to compute the simple cylinder representation for graphics. The base size is specified as the x-axis end points, the base radius, and the number of sides desired for the cylinder.

A recoverable error encountered within subroutine BASE causes an error message to be written through a call to subroutine ERRMSG followed by appropriate recovery action. There are no nonrecoverable error conditions in subroutine BASE.

Figure A2-4 is the VCLR for subroutine BASE.

IF Modification Mode					
T	Prompt for Modification Category Flag, ICHNG				Prompt for Base Location
	DOCASE ICHNG				Prompt for Base Rotation Sequence and Rotation Angles
	1	2	3	4	Def
	Modify Base Location	Modify Base Rotation Sequence and Rotation Angles	Modify Base End Points	Modify Base Radius and Number of Sides	Null
	DOUNTIL All Modifications Complete				Prompt for Base End Points
					Prompt for Base Radius and Number of Sides

Figure A2-4 BASE VCLR

2.1.1.2 Define Joint Data (JOINT) - Subroutine JOINT interactively prompts the user for the type, location, orientation, and initial state of each joint in the robotic system. The calling argument MOD specifies whether subroutine JOINT was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated allowing modification of other joint data. Upon completion of all changes, the user requests termination of modifications, and control is returned to the calling program.

The joint type is specified as either hinge, swivel, or sliding. Hinge joints rotate about the joint y axis. Swivel joints rotate about the joint x axis. Sliding joints move along the joint x axis.

The location of the joint is specified as the Cartesian coordinates of the origin of the joint coordinate system given in terms of the coordinate system of the previous joint (or base if the current joint is joint 1). Note that the x axis of a joint coordinate system is directed along the centerline of the link between that joint and the next joint (or end effector if the current joint is the final joint in the system). The orientation of the y-z plane is user-defined but is usually determined by the joint type (i.e., orientation of the axis of rotation).

The orientation of the joint is specified as a rotation sequence and corresponding set of rotation angles that define the orientation of the current joint coordinate system with respect to the coordinate system of the previous joint (or the base if the current joint is joint 1). The joint-axis orientation conventions were discussed in the preceding paragraphs.

The initial state of each joint is specified as the initial joint angle for hinge or swivel joints and the initial length for sliding joints.

A recoverable error encountered within subroutine JOINT causes an error message to be printed through a call to subroutine ERRMSG. Appropriate recovery action is then taken. There are no nonrecoverable error conditions in subroutine JOINT.

Figure A2-5 is the VCLR for subroutine JOINT.

IF Modification Mode					
	Prompt for Modification Category Flag, ICHNG				Prompt for Joint Type
	DOCASE ICHNG				Prompt for Joint Location
	1	2	3	4	Def
	Modify Joint Type	Modify Joint Location	Modify Joint Sequence and Rotation Angles	Modify Joint Variable Value	Null
	DUNTIL All Modifications Complete				Prompt for Initial Joint Variable Value

Figure A2-5 JOINT VCLR

2.1.1.3 Define Link Data (LINK) - Subroutine LINK interactively prompts the user for the size, location of the center of mass, the mass, and the inertia matrix of each link in the robotic system. The calling argument MOD specifies whether subroutine LINK was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated, allowing modification of other link data. Upon completion of all changes, the user requests termination of modifications, and control is returned to the calling program.

The size parameters the user is prompted for in subroutine LINK are used by subroutine OBJECT to compute the simple cylinder representation for graphics. The size of each link is specified as the x-axis end points, the link radius, and the number of sides desired for the cylinder.

The location of the center of mass of the link is specified as the Cartesian coordinates of the center of gravity (cg) in the coordinate system of the joint at the "base" end of the link. The location and orientation of that joint coordinate system were defined in a call to subroutine JOINT immediately preceding the current call to subroutine LINK.

The remaining link mass properties are specified by the link mass and the link inertia matrix. The inertia matrix is specified relative to the joint at the "base" end of the link.

A recoverable error encountered within subroutine LINK causes an error message to be printed through a call to subroutine ERRMSG. Appropriate recovery action is then taken. There are no nonrecoverable error conditions in subroutine LINK.

Figure A2-6 is the VCLR for subroutine LINK.

IF Modification Mode							
Prompt for Modification Category Flag, ICHNG							Prompt for Link End Points
DOCASE ICHNG							Prompt for Link Radius
1	2	3	4	5	6	Def	Prompt for Number of Sides for Link
Modify Link End Points	Modify Link Radius	Modify Number of Sides for Link	Modify Location of Center of Mass	Modify Link Mass	Modify Link Inertia Matrix	Null	Prompt for Location of Link Center of Mass
DOUNTIL All Modifications Complete							Prompt for Link Mass
							Prompt for Link Inertia Matrix

Figure A2-6 LINK VCLR

2.1.1.4 Define Tool Data (TOOL) - Subroutine TOOL interactively prompts the user for the location and orientation of the tool (or end effector) of the robotic system. The calling argument MOD specifies whether subroutine TOOL was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated, allowing modification of other tool data. Upon completion of all changes, the user requests termination of modifications, and control is returned to the calling program.

The location of the tool is specified as the Cartesian coordinates of the origin of the tool coordinate system given in terms of the coordinate system of the final joint in the robotic system. The location and orientation of the final joint was specified by the last call to subroutine JOINT.

The orientation of the tool is specified as a rotation sequence and corresponding set of rotation angles that define the orientation of the tool coordinate system with respect to the coordinate system of the final joint.

A recoverable error encountered within subroutine TOOL causes an error message to be printed through a call to subroutine ERRMSG. Appropriate recovery action is then taken. There are no nonrecoverable error conditions in subroutine TOOL.

Figure A2-7 is the VCLR for subroutine TOOL.

T	IF Modification Mode			F
Prompt for Modification Category Flag, ICHNG			Prompt for Tool Location	
DOCASE ICHNG			Prompt for Tool Rotation Sequence and Rotation Angles	
1	2	Def		
Modify Tool Location	Modify Tool Rotation Sequence and Rotation Angles	Null		
DUNTIL All Modifications Complete				

Figure A2-7 TOOL VCLR

2.1.1.5 Define Simple Graphic Representation (OBJECT) - Subroutine OBJECT creates simple cylinder graphic data used by the graphics package to draw the robotic system. The data created in subroutine OBJECT are stored in COMMON block IOBJ. The form of the data as stored in the common block is dictated by the requirements of the graphics routines. Data representing a right circular cylinder of the specified size are computed for each system component (the base and each link). For the tool (or end effector), data are computed to allow display of the tool coordinate system axes only.

Two counter arrays are used in subroutine OBJECT and stored in common to keep track of the number of components and the starting location of each within the common block. These arrays are NUM and NSTRT respectively. These arrays are used by subroutine GRAPH to locate data in the common block for the various system components.

The simple cylinder representation used by subroutine OBJECT was chosen for ease of data point computation, ease of modification, and acceptability of use as a coarse representation of most robotic system components. The use of simple cylinder representations allows the addition and deletion of links for quick study of various system configurations.

The computation of the data points defining the cylinder for the base or each link is performed in two steps. First, the data defining the vertex points around the two end circles are computed. This is accomplished by computing the y and z coordinates around the x axis at one end of the cylinder. The coordinates for the other end of the cylinder are then specified by repeating the coordinates of the first end while replacing the x value with that of the second end. The second step in defining the data for the cylinder is to define the data required to represent the sides of the cylinder by connecting corresponding vertices in the two cylinder ends.

The computation of data points to define the tool coordinate system consists of specifying pairs of points from the origin to a specified distance out each axis. The distance used for the axis length is twice the radius of the final link in the system.

Figure A2-8 is the VCLR for subroutine OBJECT.

DOCASE IN					
0		1 to N		-1	Def
	Compute Coordinates of Vertex		Compute Coordinates of Vertex	Compute Coordinates of Origin and a Reference Point on Each Axis of the Tool Coordinate System	Null
	DOUNTIL All Vertices of End Considered		DOUNTIL All Vertices of End Considered		
	DOUNTIL Each End of Base Considered		DOUNTIL Each End of Link Considered		
Select Pairs of Corresponding Vertex Data Points from Each End to Represent Side Data		Select Pairs of Corresponding Vertex Data Points from Each End to Represent Side Data			

Figure A2-8 OBJECT VCLR

2.1.1.6 System Definition Function Graphics (GRAPH) - Subroutine GRAPH provides the graphics capability in the System Definition function. Subroutine GRAPH displays the robotic system using the simple cylinder representation data computed by subroutine OBJECT. Subroutine GRAPH is called following definition of each system component (the base, each link, or the tool) and displays that component for user inspection. Capabilities exist for replacing modified components and for adding or deleting components. No provisions are made for displaying environment data within subroutine GRAPH.

Calling argument IFLAG specifies whether the base, a link, or the tool is to be displayed. The argument IFLAG also controls the initialization and termination of the graphics.

Creation and display of base data are controlled by subroutine CREATE through a sequence of calls to subroutines BASE, OBJECT, and GRAPH. If display of the base is selected, the graphics is initialized and the base is displayed in the proper position and orientation. Control is returned to subroutine CREATE where the user may elect to modify the base data through calls to subroutine BASE and OBJECT. If the base is modified, subroutine GRAPH is called to replace the current display with a display of the modified base data. The user may iterate on the base until satisfied before proceeding to definition of the system joints and links.

Creation and display of the data for each link are controlled by subroutine CREATE through a sequence of calls to subroutines JOINT, LINK, OBJECT, and GRAPH. If display of the link is selected, the new link is added to the display in the proper position and orientation. Control is returned to subroutine CREATE where the user may elect to modify the link data through calls to subroutines JOINT, LINK, and OBJECT. If the link is modified, subroutine GRAPH is called to replace the current link display with a display of the modified link data. The user may iterate on the link data until satisfied before proceeding to the next link or to the tool. The user is allowed to add or delete links even after the tool data have been defined and displayed.

Creation and display of the tool data are controlled by subroutine CREATE through a sequence of calls to subroutines TOOL, OBJECT, and GRAPH. If display of the tool is selected, the tool coordinate system is added to the display in the proper position and orientation. Control is then returned to subroutine CREATE where the user may elect to modify the tool data through calls to subroutines TOOL and OBJECT. If the tool data are modified, subroutine GRAPH is called to replace the current display of the tool data with a display of the modified tool data. The user may iterate on the tool data until satisfied before terminating the graphics display.

Subroutine GRAPH uses Evans and Sutherland graphics routines designed for use with the Evans and Sutherland Multi-Picture System.

Figure A2-9 is the VCLR for subroutine GRAPH.

0		DOCASE IFLAG				Def	
		1 to N		-1			-999
Initialize Graphics		T / Modify Link / F		T / Modify Tool / F		Terminate Graphics	Null
T / Modify Base / F		Mark Segment	Save Trans-formation Matrix	Mark Segment	Save Trans-formation Matrix		
Mark Base Segment	Null	Retrieve Trans-formation Matrix		Retrieve Trans-formation Matrix			
Open Base Segment		Open Link Segment		Open Tool Segment			
Compute Translation Matrix		Compute Translation Matrix		Compute Translation Matrix			
Compute Rotation Matrices for Rotation Sequence		Compute Joint Angle Rotation Matrix		Compute Rotation Matrices for Rotation Sequence			
Draw Component		Compute Rotation Matrices for Rotation Sequence		Draw Component			
DOUNTIL All Base Components Considered		Draw Component		DOUNTIL All Tool Components Considered			
Close Segment		Draw Component		Close Segment			
T / Modify Base / F		DOUNTIL All Link Components Considered		T / Modify Tool / F			
Replace Segment	Add Segment	Close Segment		Replace Segment	Add Segment		
		T / Modify Link / F					
		Replace Segment	Add Segment				

Figure A2-9 GRAPH VCLR

2.1.1.7 Define Desired Motion (CNTROL) - Subroutine CNTROL interactively prompts the user for coefficients of quadratic functions of time that are used to control the motion of the robotic system. The calling argument MOD specifies whether subroutine CNTROL was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated, allowing modification of other data within subroutine CNTROL. Upon completion of all changes, the user requests termination of modifications and control is returned to the calling program.

Subroutine CNTROL prompts the user for control option flag, IHIST, which indicates whether control of the system motion will be through time histories or through Evans and Sutherland analog devices. Control of the system motion through the Evans and Sutherland analog devices is not currently implemented.

For control of the system through time histories, the user may specify up to 20 time segments within which the system motion will be specified. For each time segment, the user is prompted for the segment start time, whether control is of the end-effector motion or the motion of the individual joints, and whether the time functions specify rates or positions, and the coefficients for the time functions. The use of the time functions to specify position is not currently implemented. If end-effector control is requested for the current segment, coefficients must be supplied for functions describing the end-effector translation along each axis and the end-effector rotation about each axis. If control of the individual joints is requested for the current segment, coefficients must be supplied for functions describing the motion of each joint in the system.

A recoverable error encountered within subroutine CNTROL causes an error message to be written through a call to subroutine ERRMSG followed by appropriate recovery action. There are no nonrecoverable error conditions in subroutine CNTROL.

Figure A2-10 is the VCLR for subroutine CNTROL.

T		IF Modification Mode				F				
Prompt for Modification Category Flag, ICHNG					Prompt for Motion Control Option					
DOCASE ICHNG					T Control via Time Histories F					
1		2		3		Def				
Modify Motion Control Option		Print Segment Parameters		Terminate Modifications		Null		Prompt for Segment Start Time		Null
		Modify Segment Parameters as Required						Prompt for End-Effector or Joint Control		
								Prompt for Rate or Position Control		
								T IF Joint Control F		
DOUNTIL All Segments Considered						Prompt for Joint Motion Coefficients		Prompt for End-Effector Translation Coefficients		
								Prompt for End-Effector Rotation Coefficients		
DOUNTIL All Modifications Complete					DOUNTIL All Segments Defined					

Figure A2-10 CNTROL VCLR

2.1.1.8 Specify Program Options (PRGOPT) - Subroutine PRGOPT interactively prompts the user for the program start time, stop time, time step, and several flags for the control of output and the selection of some computational capabilities. The calling argument MOD specifies whether subroutine PRGOPT was called in the create mode or the modification mode. In the create mode, the user is prompted for all possible data. In the modification mode, the user is prompted to specify which data category is to be changed. The user is then prompted only for the requested data. Following input of the requested data, the modification category prompt is repeated, allowing modification of other data within subroutine PRGOPT. Upon completion of all changes, the user requests termination of modifications, and control is returned to the calling program.

Subroutine PRGOPT allows the user to request the generation of an output file from the Analysis Tools function. The file will contain all data required by the Postprocessing function for further and more detailed study of the results of the execution of the particular analysis tool requested. The time frequency of output of data to the file is also specified.

Subroutine PRGOPT allows the user to request printed output during the Analysis Tools function execution. The content and format of the data to be printed are provided within each of the analysis tools. The flag set within subroutine PRGOPT is used only to turn on the print routines. The time frequency of the printed output is also specified.

Subroutine PRGOPT allows the user to request that dynamics computations not be performed. This flag is used within the requirements analysis tool to turn off dynamics computations to greatly speed up the execution. If the dynamics computations are turned off, no force or torque data will be available.

A recoverable error encountered within subroutine PRGOPT causes an error message to be written through a call to subroutine ERRMSG followed by appropriate recovery action. There are no nonrecoverable error conditions in subroutine PRGOPT.

Figure A2-11 is the VCLR for subroutine PRGOPT.

IF Modification Mode					
Prompt for Modification Category Flag, ICHNG			Prompt for Execution Start Time		
DOCASE ICHNG					Prompt for Execution Stop Time
1	2	3	4	Def	Prompt for Execution Time Step
Modify Start Time, Stop Time, and Time Step	Modify Output File Option Flag	Modify Print File Option Flag	Modify Dynamics Computation Flag	Null	Prompt for Output File Request Flag
DUNTIL All Modifications Complete					Prompt for Printed Output Request Flag
					Prompt for Dynamics Computations Flag

Figure A2-11 PRGOPT VCLR

2.1.2 Define Detailed Graphics Representation Data (BLDDAT)

Subroutine BLDDAT controls the program flow during the creation of the detailed graphics representation data used to display either the robotic system or the physical environment. The basic input data file generated by the System Definition function must already exist prior to calling subroutine BLDDAT. Subroutine RDSIM is called to read the input data file and load the required COMMON blocks. Subroutine DRAW is then called to initialize the graphics and draw the reference coordinate system. If subroutine BLDDAT was called to define the environment representation, the robotic system base coordinate system is used as the reference system. If the robotic system representation is being defined, the coordinate system for the appropriate section of the system is used (base, each joint/link, or tool).

Each section of the detailed representation is made up of a number of components, which are themselves simple three-dimensional solid shapes. The user is prompted for a shape type. Currently, the shapes include cylinder, cone, rectangular solid, symmetric and nonsymmetric trapezoidal solids, and a triangular cross-section beam. Additional shapes can be added as required. Based on the shape selected, an appropriate subroutine is called to prompt the user for the shape dimensions and compute the data points for that shape. Subroutine CYL is called for cylinders and cones. Subroutine RECT is called for rectangular or trapezoidal solids. Subroutine TRISTR is called for the triangular cross-section beam.

Following selection of a shape and the computation of the data points defining that shape, subroutine ORIENT is called to prompt the user for the position and orientation of the shape within the reference coordinate system. Subroutine DRAW is then called to display the component on the graphics screen for user inspection. The user may then elect to accept the component as defined or change the component. If the user elects to change the component, the shape selection prompt is reissued and the component selection and definition process is repeated. Subroutine DRAW is then called to replace the rejected component with the modified component. The user may continue to iterate on a component until satisfied. Once the user elects to accept a component, that component may no longer be modified.

After the component is satisfactory, subroutine DBASE is called to add that component to the data defining the representation of the environment or of the section of the robotic system under consideration. The user then specifies through prompt response whether another component is to be defined. When defining the environment representation, components are defined until the entire environment representation is complete. When defining the robotic system representation, each major section of the system is defined in turn (base, each link, tool). Components for each section are defined until that section is complete before continuing to the next section.

After all input is complete, subroutine DRAW is called to terminate the graphics. Subroutine WRTSIM is then called to write the new data file containing the detailed graphics representation data.

A recoverable error encountered within subroutine BLDDAT causes an error message to be written through a call to subroutine ERRMSG. Appropriate recovery action is then taken. Nonrecoverable errors encountered within routines called by BLDDAT cause return of program control through the System Definition function routines to the ROBSIM executive program.

Figure A2-12 is the VCLR for subroutine BLDDAT.

CALL RDSIM to Read Input File						
T	If Defining Environment Graphics Data					F
NLNK = -1			Null			
CALL DRAW to Draw Reference Coordinate System						
Increment Component Counter and Zero Change Flag						
Prompt for Component Shape Flag, ISHAPE						
DOCASE ISHAPE						
1	2	3	4	5	6	Def
CALL CYL for Cylinder	CALL CYL for Cone	CALL RECT for Rectangular Solid	CALL RECT for Symmetric Trapezoidal Solid	CALL RECT for Non- Symmetric Trapezoidal Solid	CALL TRISTR for Triangular Beam	Null
CALL ORIENT to Locate and Orient Component						
CALL DRAW to Display Component						
DOUNTIL Component Is Correct						
CALL DBASE to Add Component to Data File						
DOUNTIL All Components Have Been Defined						
DOUNTIL Environment or Entire Robotic System Graphics Data Definition Is Complete						
CALL DRAW to Terminate Graphics						
CALL WRTSIM to Write New Data File to Disk						

Figure A2-12 BLDDAT VCLR

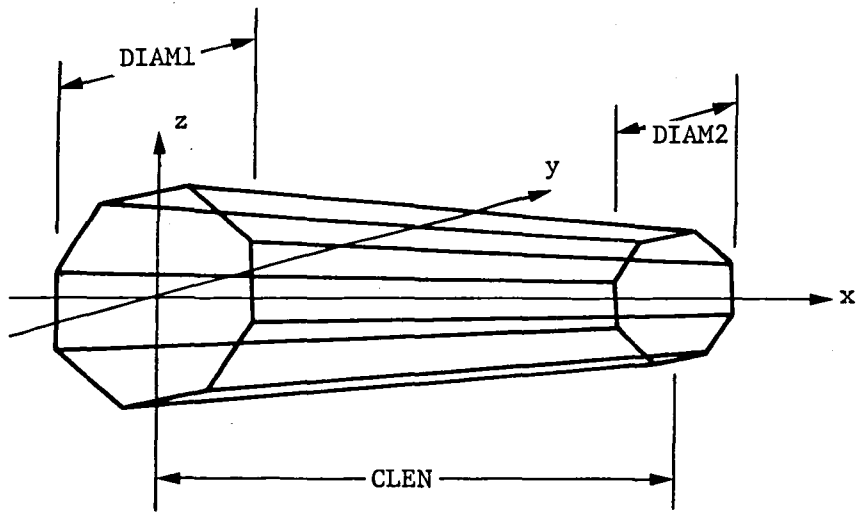
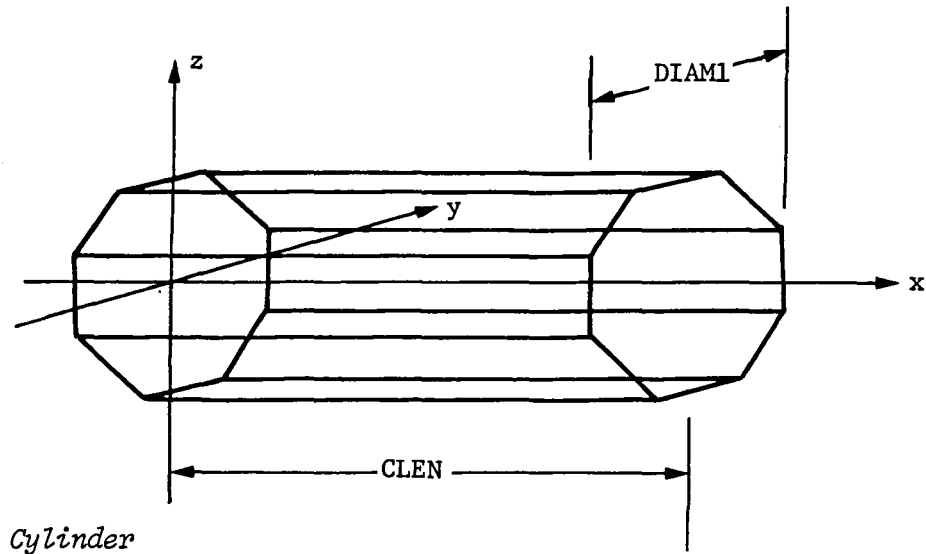
2.1.2.1 Define Cylinder Shape (CYL) - Subroutine CYL is called from subroutine BLDDAT during the definition of detailed graphic representations for the environment or the robotic system. Subroutine CYL is called if the requested component is a cylinder or a cone.

The user is prompted for the shape diameter, DIAM1 (diameter for a cylinder, base diameter for a cone). If the shape is a cone, the user is then prompted for the top diameter of the cone, DIAM2. The user is then prompted for the shape length, CLEN.

Using the shape size parameters, the data points describing the shape are computed. The data defining the shape are computed in a shape coordinate system. Figure A2-13 shows the coordinate systems used for cylinders and cones. For a cylinder or cone, the x axis is along the shape centerline. The shape length is measured from $x = 0$ to $x = \text{CLEN}$. The orientation of the y-z plane is arbitrary for the cylinder or cone shapes. The vertex points for the shape base (at $x = 0$) are computed using eight sides and the shape diameter DIAM1. The vertex points for the shape end at $x = \text{CLEN}$ are computed using eight sides and either DIAM1 for a cylinder or DIAM2 for a cone. The data points used to define the shape sides are computed by selecting pairs of corresponding vertex points.

The data computed by subroutine CYL are stored in the array ARRAY as Cartesian coordinates in the shape coordinate system. The counter N1 contains the number of points describing the two shape ends. These points are connected sequentially by the graphics routines. The counter N2 contains the number of points describing the shape sides. These points are connected in alternating pairs by the graphics routines.

Figure A2-14 is the VCLR for subroutine CYL.



Cone

Figure A2-13 Cylinder and Cone Shape Coordinate Systems

Prompt for Cylinder Diameter (or Cone Diameter at $x = 0$), DIAM1					
T	IF Cone Requested			F	
Prompt for Cone Diameter at $X = \text{CLEN}$, DIAM2			Null		
Prompt for Shape Length, CLEN					
	Set End Radius to $\text{DIAM1}/2.0$				
	T	IF Second End and Cone			F
	Set End Radius to $\text{DIAM2}/2.0$			Null	
	Compute y and z Location of Vertex				
	T	IF Second End			F
	X = CLEN		X = 0.0		
	Store Current Vertex Coordinates in ARRAY				
	DOUNTIL All Vertices Computed				
	DOUNTIL Both Ends Considered				
	Store in ARRAY Pairs of Corresponding Vertex Points from Each End to Define Shape Side Data				
	DOUNTIL All Sides Considered				

Figure A2-14 CYL VCLR

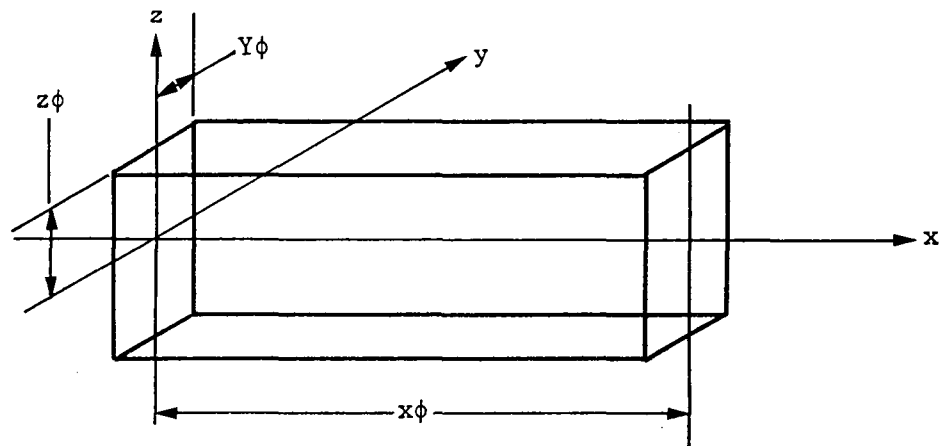
2.1.2.2 Define Rectangular Shape (RECT) - Subroutine RECT is called from subroutine BLDDAT during the definition of detailed graphic representations for the environment or the robotic system. Subroutine RECT is called if the requested component is a rectangular solid, a symmetric trapezoidal solid, or a nonsymmetric trapezoidal solid.

The user is prompted for the shape length (x-axis dimension, X0), the shape width (+y-axis dimension, Y0), and the appropriate z-axis dimensions. For a rectangular solid, the user is prompted for the shape height (+z-axis dimension, Z0). For a symmetric trapezoidal solid, the user is prompted for the height of each end of the trapezoid (x = 0 end z-axis dimension, Z1; and x = X0 end z-axis dimension, Z2). For a nonsymmetric trapezoidal solid, the user is prompted for the height of each end of the trapezoid (x = 0 end z-axis dimension, Z1, and x = X0 end z-axis dimension, Z2).

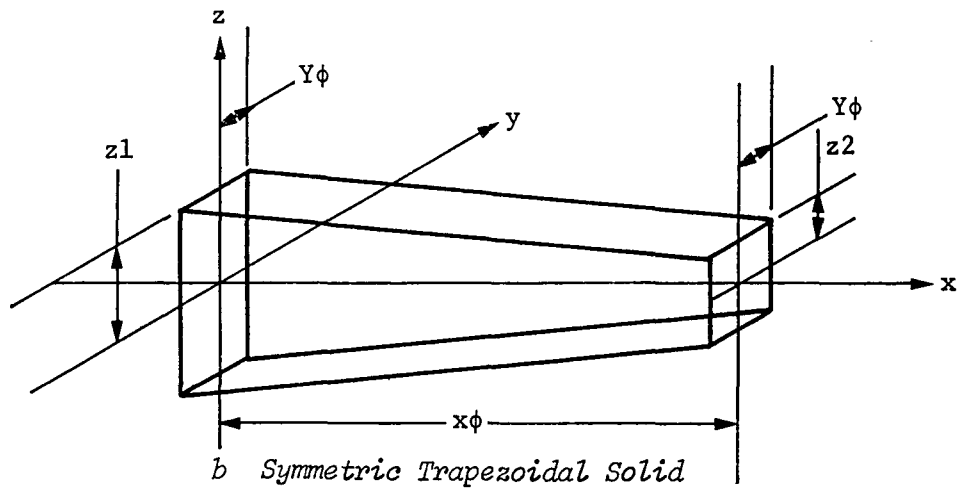
Using the shape size parameters, the data points describing the shape are computed. The data defining the shape are computed in a shape coordinate system. Figure A2-15 shows the coordinate systems used for rectangular or trapezoidal solids. The x axis is along the shape centerline. The shape length is measured from x = 0 to x = X0. The rectangular and symmetric trapezoidal solid shapes are symmetric about both the y axis and the z axis. The nonsymmetric trapezoidal solid is symmetric about the y axis but not the z-axis. The vertex points for the shape base (at x = 0) are computed using +Y0 and either +Z0, +Z1, or 0 and -Z1 for the rectangle, the symmetric trapezoid, or the nonsymmetric trapezoid, respectively. The vertex points for the shape end at x = X0 are computed using +Y0 and either +Z0, +Z2, or 0 and -Z2 for the rectangle, the symmetric trapezoid, or the nonsymmetric trapezoid, respectively. The data points used to define the shape sides are computed by selecting pairs of corresponding vertex points.

The data computed by subroutine RECT are stored in the array ARRAY as Cartesian coordinates in the shape coordinate system. The counter N1 contains the number of points describing the two shape ends. These points are connected sequentially by the graphics routines. The counter N2 contains the number of points describing the shape sides. These points are connected in alternating pairs by the graphics routines.

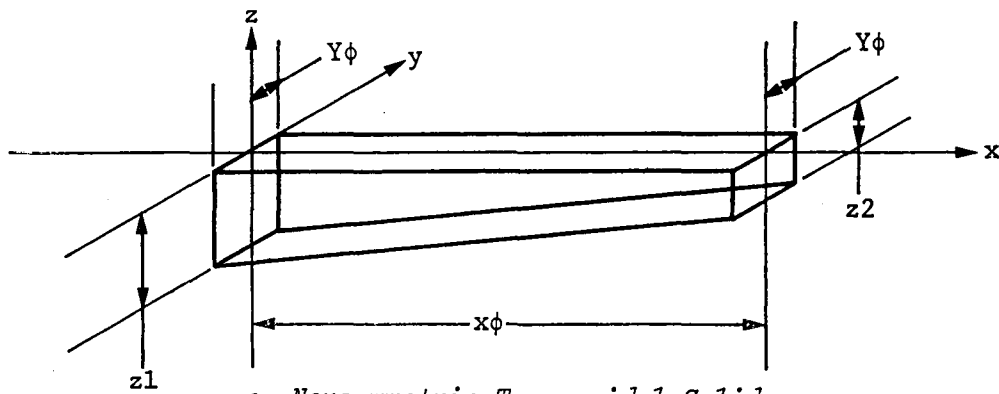
Figure A2-16 is the VCLR for subroutine RECT.



a Rectangular Solid



b Symmetric Trapezoidal Solid



c Nonsymmetric Trapezoidal Solid

Figure A2-15 Rectangular and Trapezoidal Shape Coordinate Systems

Prompt for Shape Length, $x\phi$	
Prompt for Shape Width, $y\phi$	
T	IF Shape is Rectangular Solid
Prompt for Shape Height, $z\phi$	Prompt for Trapezoid Height at $x = 0, z1$
	Prompt for Trapezoid Height at $x = x\phi, z2$
Using Shape Size Parameters, Compute Coordinates of Each Vertex	
DUNTIL Both Ends Considered	
Select Pairs of Corresponding Vertex Points from Each End to Define Shape Side Data	
DUNTIL All Sides Considered	

Figure A2-16 RECT VCLR

2.1.2.3 Define Triangular Beam Shape (TRISTR) - Subroutine TRISTR is called from subroutine BLDDAT during the definition of detailed graphic representations for the environment or the robotic system. Subroutine TRISTR is called if the requested component is a triangular cross-section beam.

The user is prompted for the length of the base (also used as the triangle height) of the triangular cross section of the beam, TRIL. The user is then prompted for the length of a segment of the beam, SEGL. Finally, the user is prompted for the number of segments in the beam, NSEG.

Using the beam size parameters, the data points describing the beam are computed. The data defining the beam are computed in a shape coordinate system. Figure A2-17 shows the coordinate system used for beams. The beam length is measured along the x axis from $x = 0$ to $x = \text{NSEG} * \text{SEGL}$. One side of the triangular cross section lies on the y axis from $y = -\text{TRIL}/2.0$ to $y = +\text{TRIL}/2.0$. One vertex of the triangular cross section lies on the line defined by $x = 0$ to $x = \text{NSEG} * \text{SEGL}$, $y = 0$, and $z = -\text{TRIL}$. The vertex points for a triangle are computed at the base ($x = 0$) of the beam and at the end of each segment along the beam length. The data points used to define the beam sides are computed by selecting pairs of corresponding vertex points from the two ends of the beam.

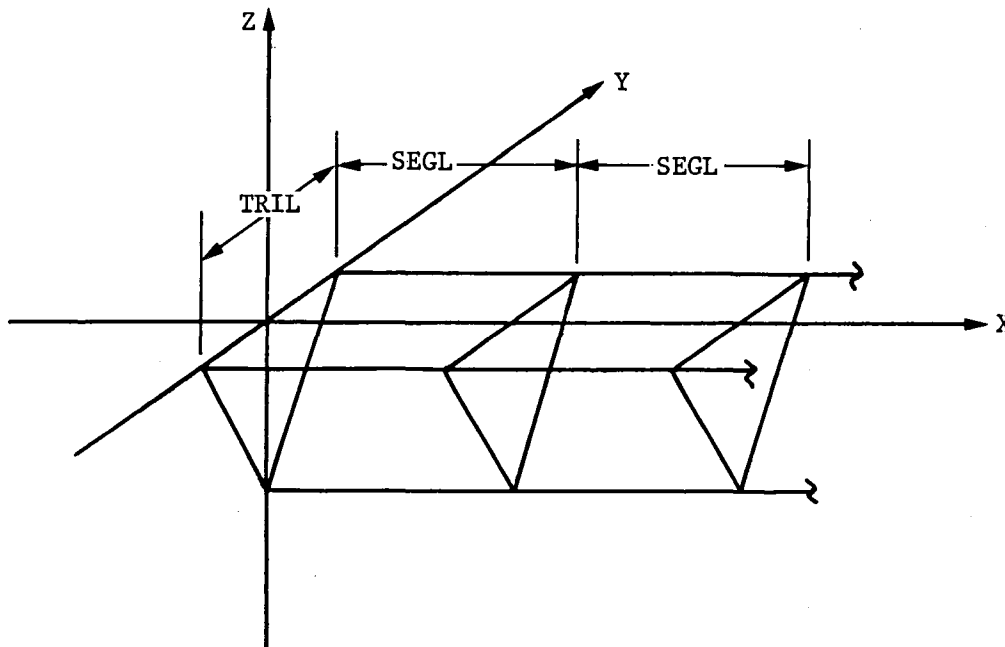


Figure A2-17 Triangular Cross-Section Beam Coordinate System

The data computed by subroutine TRISTR are stored in ARRAY as Cartesian coordinates in the shape coordinate system. The counter N1 contains the number of points describing the triangles along the beam and at each beam end. These points are connected sequentially by the graphics routines. The counter N2 contains the number of points describing the beam sides. These points are connected in alternating pairs by the graphics routines.

Figure A2-18 is the VCLR for subroutine TRISTR.

Prompt for Length of Side of Triangular Cross Section	
Prompt for Length of Segment of Beam	
Prompt for Number of Segments in Beam	
	Compute Coordinates of Vertices of Triangle at Segment End
DOUNTIL All Segments Considered	
	Select Pairs of Corresponding Vertex Points from Each End of the Beam to Define the Beam Sides
DOUNTIL All Sides Considered	

Figure A2-18 TRISTR VCLR

2.1.2.4 Orient Component (ORIENT) - Subroutine ORIENT is called from subroutine BLDDAT during the definition of detailed graphic representations for the environment or the robotic system. Subroutine ORIENT is called to position the current component properly within the reference coordinate system.

The user can request input of a rotation sequence via prompt response. If rotations are required, the user is prompted for an axis of rotation and corresponding rotation angle for each desired rotation. After all rotations have been defined, subroutine MAT is called to compute the total rotation transformation matrix corresponding to the requested rotation sequence. Each set of coordinates in ARRAY is then transformed from the shape coordinate system to the reference coordinate system through calls to subroutine MATVEC using the rotation transformation matrix.

The user is then prompted for a translation vector to position the origin of the component within the reference coordinate system. This translation vector is then added to each set of coordinates in ARRAY.

Figure A2-19 is the VCLR for subroutine ORIENT.

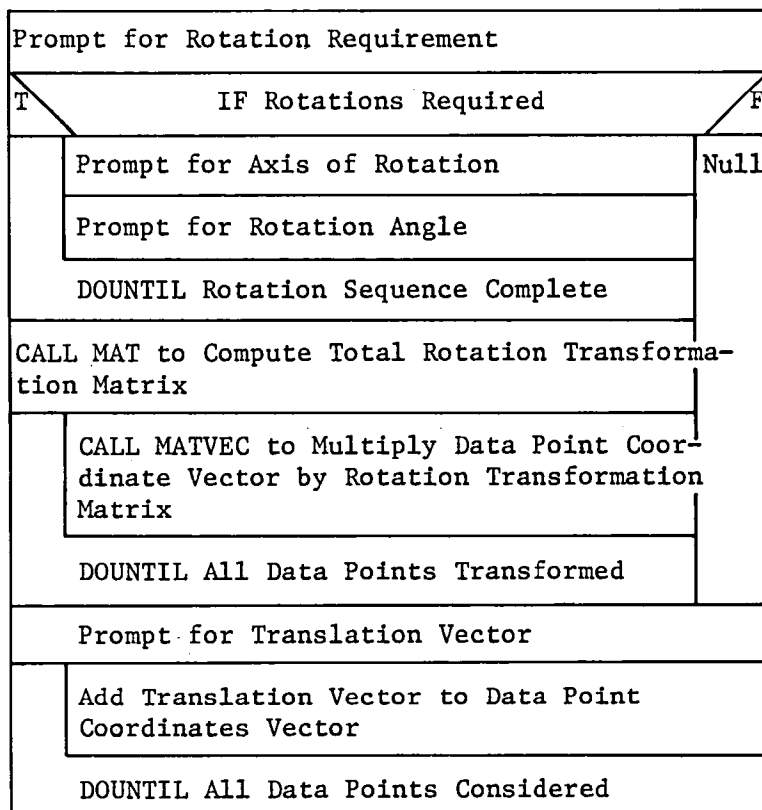


Figure A2-19 ORIENT VCLR

2.1.2.4.1 Rotation Matrix (MAT) - Subroutine MAT is called from subroutine ORIENT to compute the total rotation transformation matrix defined by the input rotation sequence and angles.

The rotation sequence and corresponding angles input from subroutine ORIENT describe rotations from the reference system to the desired component system. The transformation matrix desired from subroutine MAT is from the component system to the reference system. Therefore, the transpose (inverse) of the normal x, y, and z-axis rotation matrices are used.

For each rotation in the input rotation sequence, the axis rotation matrix is loaded and premultiplied with the current total transformation matrix. The axis rotation matrices used are:

For x-axis rotation

1.0	0.0	0.0
0.0	cos(ang)	-sin(ang)
0.0	sin(ang)	cos(ang)

For y-axis rotation

cos(ang)	0.0	sin(ang)
0.0	1.0	0.0
-sin(ang)	0.0	cos(ang)

For z-axis rotation

cos(ang)	-sin(ang)	0.0
sin(ang)	cos(ang)	0.0
0.0	0.0	1.0

Figure A2-20 is the VCLR for subroutine MAT.

DOCASE Rotation Axis			
1	2	3	Def
Compute Rotation Matrix about x-Axis	Compute Rotation Matrix about y-Axis	Compute Rotation Matrix about z-Axis	Null
Compute Updated Total Transformation Matrix			
DUNTIL Entire Rotation Sequence Considered			

Figure A2-20 MAT VCLR

2.1.2.4.2 Matrix/Vector Multiplication (MATVEC) - Subroutine MATVEC is called from subroutine ORIENT to provide matrix/ vector multiplication. Input Vector A is multiplied by input matrix TRANS to produce output Vector B.

$$\begin{bmatrix} B(1) \\ B(2) \\ B(3) \end{bmatrix} = \begin{bmatrix} TRANS(1,1) & TRANS(1,2) & TRANS(1,3) \\ TRANS(2,1) & TRANS(2,2) & TRANS(2,3) \\ TRANS(3,1) & TRANS(3,2) & TRANS(3,3) \end{bmatrix} \begin{bmatrix} A(1) \\ A(2) \\ A(3) \end{bmatrix}$$

Figure A2-21 is the VCLR for subroutine MATVEC.

Multiply Input Vector by Input Matrix

Figure A2-21 MATVEC VCLR

2.1.2.5 Draw Component (DRAW) - Subroutine DRAW is called from subroutine BLDDAT to provide graphics display during the definition of detailed graphic representations for the environment or the robotic system. Subroutine DRAW is called to display each successive component as it is defined.

Input argument IFLAG controls the routine logic. A value of IFLAG = 1 indicates the start of a new definition section. Sections are specified by input argument M. If the current section is the first section considered (M = 1), then the graphics system is initialized and subroutine ESMAT is called to compute transformation matrices for all sections that will be required. The reference coordinate system for the current section is then drawn.

A value of IFLAG = 2 indicates that the component under consideration be displayed for the first time. The current component number is specified by input argument NC. A new display segment is opened, the component is drawn, and the segment is added to the current display.

A value of IFLAG = 3 indicates that the component under consideration has been modified and the modified version is to be displayed. The component display segment is marked for update, the segment opened, and the component drawn. The segment is then replaced in the current display.

A value of IFLAG = 4 indicates that all display is complete and termination of the graphics is requested.

Figure A2-22 is the VCLR for subroutine DRAW.

1		DOCASE IFLAG			Def
T	IF First Call	2	3	4	
Initialize Graphics with E&S Routines	Null	Retrieve Transformation Matrix for Current Link	Retrieve Transformation Matrix for Current Link	Terminate Graphics with E&S Routine MPINIT	Null
CALL ESMAT for Transformation Matrix		Open Segment for Current Component	Mark Component Segment for Update		
Store Matrix		Set Color	Set Color		
		CALL D3DATA to Draw Component	CALL D3DATA to Draw Component		
DOUNTIL all Links Considered		Close Segment	Close Segment		
Draw Reference Coordinate System		Add Segment	Replace Segment		

Figure A2-22 DRAW VCLR

2.1.2.5.1 Graphics Transformation Matrix (ESMAT) - Subroutine ESMAT is called from subroutine DRAW to compute the transformation matrix from each system section coordinate system to the graphics coordinate system. The environment data have only one section. The environment is defined in the robotic system base coordinate system. The robotic system has section coordinate systems for the base, each joint/link, and the end effector.

Subroutine ESMAT uses Evans and Sutherland graphics routines to construct the required transformation matrices. Input argument K specifies which system section is under consideration.

A value of $K = 1$ indicates the robotic system base or the environment. The transformation matrix is composed of a translation matrix based on the base location and rotation matrices constructed using the base orientation parameters.

A value of K from 2 to the number of links in the system (N) plus 1 (N+1) indicates the (K-1)th joint/link. The transformation matrix computed by each call to subroutine ESMAT is automatically concatenated to the previous matrix, thereby forming the total transformation from the current section coordinate system to the graphics coordinate system. The joint/link transformation matrix is composed of a translation matrix based on the joint position, a rotation matrix based on the initial joint angular displacement, and rotation matrices constructed using the joint orientation parameters. Joint position and orientation are specified relative to the coordinate system of the previous joint (base if the current joint is the first joint in the system).

A value of $K = N+2$ indicates the end-effector system. The transformation matrix for the end-effector is composed of a translation matrix based on the end-effector position and rotation matrices constructed from the end-effector orientation parameters. The end-effector location and orientation are specified relative to the coordinate system of the final joint in the system.

Figure A2-23 is the VCLR for subroutine ESMAT.

DOCASE K			
1	2	3	Def
Compute Translation Matrix for Base	Compute Translation Matrix for Current Link	Compute Translation Matrix for Tool	Null
Decode Rotation Axis Sequence	Convert Current Joint Angle to Graphics Units	Decode Rotation Axis Sequence	
Convert Rotation Angles to Graphics Units	Compute Rotation Matrix for Joint Angle	Convert Rotation Angles to Graphics Units	
Compute Rotation Matrix for Third Rotation of Sequence	Decode Rotation Axis Sequence	Compute Rotation Matrix for Third Rotation of Sequence	
Compute Rotation Matrix for Second Rotation of Sequence	Convert Rotation Angles to Graphics Units	Compute Rotation Matrix for Second Rotation of Sequence	
Compute Rotation Matrix for First Rotation of Sequence	Compute Rotation Matrix for Third Rotation of Sequence	Compute Rotation Matrix for First Rotation of Sequence	
Retrieve Total Translation-Rotation Transformation Matrix	Compute Rotation Matrix for Second Rotation of Sequence	Retrieve Total Translation-Rotation Transformation Matrix	
	Compute Rotation Matrix for First Rotation of Sequence		
	Retrieve Total Translation-Rotation Transformation Matrix		

Figure A2-23 ESMAT VCLR

2.1.2.6 Load Graphics Data to COMMON (DBASE) - Subroutine DBASE is called from subroutine BLDDAT during definition of detailed graphics representations for the environment or the robotic system. Subroutine DBASE is called to load the data for a component into the appropriate graphics data COMMON. Input argument IMAN specifies whether the component data are part of the environment or part of the robotic system.

If the component is part of the robotic system, the data are loaded into COMMON/IOBJ/. The parameter NUM contains the number of components that make up the current robotic system section. The parameter NSTRT contains the locations within COMMON/IOBJ/ at which the data for each component start.

If the component is part of the environment, the data are loaded into COMMON/ENVIRN/. The parameter NUME contains the number of components that make up the environment data.

The manner in which the data are stored in the COMMON blocks is dictated by the data format used in Evans and Sutherland graphics routine D3DATA. It should be noted that the graphics routines require INTEGER*2 data.

Figure A2-24 is the VCLR for subroutine DBASE.

Set Size Factor Parameter	
T If Defining Environment Graphics Data F	
Update Number of Components Counter for Environment	Update Number of Components Counter for Current Link
Load into COMMON/ENVIRN/ All Data Points for Current Component That Are to be Connected Consecutively by Graphics	Store Starting Location of Current Component Data within COMMON/IOBJ/
	Load into COMMON/IOBJ/ All Data Points for Current Component That Are to be Connected Consecutively by Graphics
	Load into COMMON/IOBJ/ All Data Points for Current Component That Are to be Connected in Alternating Pairs by Graphics

Figure A2-24 DBASE VCLR

2.2 ANALYSIS TOOLS FUNCTION DRIVER (SIMDRVR)

The Analysis Tools function driver operates in an interactive mode and prompts the user for the analysis tool option desired. Currently valid options are (1) requirements analysis without graphics; (2) requirements analysis with graphics; and (3) return to the ROBSIM executive.

The requirements analysis tool is the only analysis tool currently implemented within the ROBSIM program. The simulation tool has been developed outside the ROBSIM framework and is documented in Section 3.0 of this appendix. Additional analysis tools will be added as program capabilities are expanded.

If Option 2 is selected, the system motion is displayed during program execution. An output file of joint variable time histories can be generated that allows replay of the system motion in the Postprocessing function.

The requirements analysis tool allows the user to specify a desired system motion and computes the forces and torques necessary to produce that motion. The motion may be specified as individual joint rates or as desired end-effector rates. If end-effector rates are given, the requirements analysis tool computes the corresponding individual joint rates required.

If either Option 1 or 2 is selected, subroutine RDSIM is called to read the input file created by the System Definition function. Subroutine REQUIR is then called to perform the requirements analysis.

A recoverable error encountered within subroutine SIMDRVR causes an error message to be written through a call to subroutine ERRMSG. Appropriate recovery action is then taken. Nonrecoverable errors encountered within routines called by SIMDRVR cause return of control to the ROBSIM executive program.

Figure A2-25 is the VCLR for subroutine SIMDRVR.

Prompt for Analysis Tools Function Mode, IMODE			
DOCASE IMODE			
1	2	3	Def
CALL RDSIM to Read Input File	CALL RDSIM to Read Input File	Return to ROBSIM Executive	Null
CALL REQUIR to Perform Requirements Analysis without Graphics	CALL REQUIR to Perform Requirements Analysis with Graphics		

Figure A2-25 SIMDRVR VCLR

2.2.1 Requirements Analysis Tool (REQUIR)

Subroutine REQUIR controls the program logic flow for the requirements analysis capability. Subroutine REQUIR is called from subroutine SIMDRVR within the Analysis Tools function.

If printed output is desired, the user is prompted for the name of the file to be opened for printed output. The requested file is then opened. If dynamics computations are requested, the user is prompted for the name of the file to be opened for the dynamics output. The requested file is then opened. If a data file for postprocessing is desired, the user is prompted for the name of the file to be opened for postprocessing data. The requested file is then opened. Subroutine LDCOM is then called to load the COMMON parameters required by subroutine CNTRLR, which handles the motion and dynamics computations. If graphics are requested, subroutine GRAFIX is called to initialize the graphics system and the display.

Subroutine REQUIR executes a time loop from the specified start time to the specified stop time using the user-requested time increment. Within the time loop, subroutine SEGMNT is called to load the proper coefficients specifying the required system motion. Subroutine CNTRLR is called to compute the system motion and dynamics for each time step. The calls to the graphics routines are handled by subroutine CNTRLR, if required.

After completion of the time loop, any open files are closed. If graphics were requested, subroutine GRAFIX is called to terminate the graphics display.

Figure A2-26 is the VCLR for subroutine REQUIR.

T	Printed Output Requested	F
	Prompt for File Name for Printed Output File	Null
	Open Printed Output File	
T	Dynamics Computations Requested	F
	Prompt for File Name for Dynamics Output File	Null
	Open Dynamics Output File	
T	Data File for Postprocessing Requested	F
	Prompt for File Name for Postprocessing Data File	Null
	Open Postprocessing Data Output File	
CALL LDCOM to Interface Common Structure for CNTRLR		
T	Graphics Requested	F
	Initialize Graphics and Display	Null
Set Time to Start Time		
Time Equals Time Plus Time Step		
T	Motion Control via Time Histories	F
	CALL SEGMNT to Load Proper Motion Coefficients	Null
	CALL CNTRLR to Compute System Motion from Time Histories	
DOUNTIL Stop Time		
T	Printed Output Requested	F
	Close Printed Output File	Null
T	Dynamics Computations Requested	F
	Close Dynamics Output File	Null
T	Data File for Postprocessing Requested	F
	Close Postprocessing Data Output File	Null
T	Graphics Requested	F
	Terminate Graphics	Null

Figure A2-26 REQUIR VCLR

2.2.1.1 Load COMMON (LDCOM) - Subroutine LDCOM is called from subroutine REQUIR to load COMMON parameters used by subroutine CNTRLR, which handles motion and dynamics computations within the requirements analysis tool. The data considered in subroutine LDCOM come from the COMMON structures used to store the data input from the file created in the System Definition function.

The data include joint type, location, and initial joint angular displacement. Some program option flags and similar data are also loaded.

Figure A2-27 is the VCLR for subroutine LDCOM.

	Load Joint Location Array
	Load Initial Joint Variable Array
	Load Joint Type Array
	DOUNTIL All Joints Considered
	Load End-Effector Location Array
	Load Number of Joints Variable
	Load Logical Units Variables
	Load Output Options Flags
	Load Stop Time and Time Step Variables

Figure A2-27 LDCOM VCLR

2.2.1.2 System Motion Graphics (GRAFIX) - Subroutine GRAFIX provides the motion graphics capability in the Analysis Tools and Postprocessing functions. Subroutine GRAFIX displays the environment and the robotic system motion within the environment. Input argument IFLAG controls the logic flow in subroutine GRAFIX.

If IFLAG = 1, the graphics system is initialized and the environment and robotic system are displayed in the initial condition. If IFLAG = 2, the display is updated to the current time step condition. In the update mode, the environment and the system base are constant and therefore, are not updated. If IFLAG = 3, the motion is complete and the graphics display is terminated.

Evans and Sutherland graphics routines are used to provide all graphics capabilities.

Figure A2-28 is the VCLR for subroutine GRAFIX.

T	Update System Display (IFLAG = 2)		F
Mark and Open System Segment			Null
T	Initialize Display (IFLAG = 1)		F
Initialize Graphics			Null
Open Base Segment			
Compute Translation Matrix for Base			
Compute Rotation Matrices for Base Rotation Sequence			
Prompt for Environment Data Display			
T	Display Environment		
	Draw Component	Null	
DOUNTIL All Environment Components Drawn			
	Draw Component		
DOUNTIL All Base Components Drawn			
Close Base Segment			
Add Base Segment to Display			
Open System Segment			
	Open Link Segment		
	Close Link Segment		
DOUNTIL All Link Segments Nested in System Segment			
Close System Segment			
Add System Segment			
Mark System Segment for Update			
Open System Segment			

Figure A2-28 GRAFIX VCLR

T Initialize or Update Display (IFLAG = 1 or 2)		F
	Open Link Segment	Null
	Compute Translation Matrix for Link	
	Compute Joint Angle Rotation Matrix	
T Initial Pass (IFLAG = 1)		F
	CALL MATINT to Compute and Store Rotation Matrices for Link Rotation Sequence	Retrieve Rotation Matrices for Link Rotation Sequence
	Compute Total Transformation Matrix for Link	
	Draw Component	
DOUNTIL All Link Components are Drawn		
	Close Link Segment	
DOUNTIL All Links Displayed		
Open Tool Segment		
T Initial Pass (IFLAG = 1)		F
	CALL MATINT to Compute and Store Rotation Matrices for Tool Rotation Sequence	Retrieve Rotation Matrices for Tool Rotation Sequence
	Compute Total Transformation Matrix for Tool	
	Draw Component	
DOUNTIL All Tool Components are Drawn		
Close Tool Segment		
Close System Segment		
Replace System Segment in Display		
T Terminate Graphics (IFLAG = 3)		F
Terminate Graphics		Null

Figure A2-28 (concluded)

Graphics Matrix Initialization (MATINT) - Subroutine MATINT is called from subroutine GRAFIX to provide initial computation of certain constant matrices, which are then stored for later use by subroutine GRAFIX. The computation and storage of the matrices decrease execution time for system display.

Subroutine MATINT is called once for each joint/link in the robotic system and once for the end effector. In each case, the matrix representing the total of the translation and the rotation sequence defining the location and orientation of that part of the robotic system is computed.

Evans and Sutherland graphics routines are used to perform the matrix computations and concatenations.

Figure A2-29 is the VCLR for subroutine MATINT.

T	Link Data Requested		F
	Decode Link Rotation Sequence	Compute Translation Matrix for Tool	
Convert Rotation Angles to Graphics Units		Decode Tool Rotation Sequence	
Compute Rotation Matrix for Third Rotation of Link Rotation Sequence		Convert Rotation Angles to Graphics Units	
Compute Rotation Matrix for Second Rotation of Link Rotation Sequence		Compute Rotation Matrix for Third Rotation of Tool Rotation Sequence	
Compute Rotation Matrix for First Rotation of Link Rotation Sequence		Compute Rotation Matrix for Second Rotation of Tool Rotation Sequence	
Retrieve Total Link Rotation Transformation Matrix		Compute Rotation Matrix for First Rotation of Tool Rotation Sequence	
		Retrieve Total Tool Translation - Rotation Transformation Matrix	

Figure A2-29 MATINT VCLR

2.2.1.3 Load Time Segment Coefficients (SEGMNT) - Subroutine SEGMNT loads the appropriate coefficients for the time functions that define the desired system motion. The coefficients were specified during the System Definition function. A maximum of 20 time segments is allowed with coefficients specified for each segment. The coefficients are stored in COMMON/CONTRL/. The coefficients required for the current time segment are selected from the stored data and loaded into the current working array VELPRO.

At each time step, subroutine SEGMNT is called to ensure that the proper coefficients are loaded. If a new segment has begun, the proper coefficients are loaded from COMMON/CONTRL/ to array VELPRO.

Figure A2-30 is the VCLR for subroutine SEGMNT.

T / First Call to SEGMNT / F	
Null	T / Current Segment Not Final Segment / F
	T / Time Not within Current Segment / F Null
Load Motion Coefficients for Current Time Segment	Null

Figure A2-30 SEGMNT VCLR

2.2.1.4 Requirements Function Motion Controller (CNTRLR) - Subroutine CNTRLR controls the motion of manipulator models during requirements analysis. CNTRLR is called at every time step and supervises computations that describe both the position and velocity of manipulator components during requirements analysis. CNTRLR also operates on user input to activate the following ROBSIM program options:

- 1) Print out a detailed file describing the motion of the manipulator model;
- 2) Perform dynamics computations to determine forces and torques acting at the joints of the model as a result of input motion;
- 3) Display a graphics representation of the manipulator model during motion;
- 4) Save a file describing the motion of the model for later graphics display.

CNTRLR can use either of two methods of motion control, depending on the user's selection:

- 1) Individual Joint Rate Control - The user inputs a rate profile for each joint that specifies the rotation rate of the joint as a function of time. Using this form of motion control, CNTRLR directly computes the rotation rate of each joint.
- 2) Coordinated End Effector (Tool) Control - The user inputs a set of six rate profiles that specify tool velocity as a function of time. Using this form of motion control, CNTRLR first computes the specified tool velocity, then calculates the set of individual joint rates that yield the best approximation to the desired tool velocity.

Figure A2-31 is the VCLR for subroutine CNTRLR.

Set up Internal Constants		
<div><div><div>T</div><div>Dynamics Computation Requested? (IDYN = 1?)</div><div>F</div></div></div>		
Save Values of Joint Displacements and Velocities from Current Time Step		Null
Call SETUP: Calculate Control Variables for Current Time Step		
<div><div><div>T</div><div>Individual Joint Control? (ICNTRL = 2?)</div><div>F</div></div></div>		
Compute Joint Rates from Individual Joint Rate Profiles		Call INVERT: Invert Control Matrix and Compute Individual Joint Rates from End-Effector Rates
Call CHLIM: Check Velocity and Acceleration Limits (Not Yet Implemented)		
<div><div><div>T</div><div>Hard Copy Requested? (IPRT > 1?)</div><div>F</div></div></div>		
Call POUTC: Perform Additional Calculations for Printout		Null
Call WROUT: Write to Print File		
<div><div><div>T</div><div>Dynamics Computation Requested and at Least One Time Step Completed? (IDYN = 1 and I > 1?)</div><div>F</div></div></div>		
Call DYNAMICS: Perform Dynamics Computations		Null
Do While $1 < IC \leq N$ (Do for Each Joint)		
Increment Joint Displacements for Next Time Step		
Reduce Joint Displacements Modulo 2π		
Set Joint Variable for Graphics Display		
<div><div><div>T</div><div>Simulation File Requested? (ISIM = 1?)</div><div>F</div></div></div>		
Write to Simulation File		Null
<div><div><div>T</div><div>Graphics Requested? (IMODE = 2)</div><div>F</div></div></div>		
Set Parameters for Grafix		Null
Call GRAFIX: Execute Graphics Display		

Figure A2-31 CNTRLR VCLR

2.2.1.4.1 Set Up Computations (SETUP) - SETUP is called at each time step during requirements analysis and computes quantities needed in other subroutines under CNTRLR. Hence, SETUP acts to "set up" succeeding calculations.

The quantities computed by SETUP include the following:

- 1) The transformation matrices between neighboring joint coordinate systems and between joint and base coordinate systems;
- 2) The vectors describing tool location with respect to each joint (the vectors are expressed in joint coordinates);
- 3) Transformation of vectors representing links into base coordinates;
- 4) The control matrix used to compute tool velocity given individual joint rates (see Appendix D);
- 5) The specified tool velocity from the input rate profiles (only performed when coordinated tool rate control is specified).

Figure A2-32 is the VCLR for subroutine SETUP.

Call CCTM: Compute Transformation Matrices for this Time Step	
Compute Joint-to-Tool Vectors in Joint Coordinates	
Transform Link Vectors to Inertial Coordinates	
Calculate Control Matrix	
T	Individual Joint Control? (ICNTRL = 2 ?)
	F
Null	Compute Tool Velocity from Rate Profiles
T	First Time Step? (TIME = 0 ?)
	F
Compute Initial Position of Tool in Inertial Coordinates	Null

Figure A2-32 SETUP VCLR

2.2.1.4.2 Perform Output Calculations (POUTC) - POUTC is only called if a detailed printout of manipulator model motion data is requested by the user. If such a request has been made, POUTC performs at each time step computations of quantities that appear in the printout but are not required elsewhere in the ROBSIM program. The quantities computed by POUTC are:

- 1) Tool position in base coordinates at the end of the time step;
- 2) Transformation of all joint-to-tool vectors to base coordinates;
- 3) Ideal tool position (coordinated tool rate control only);
- 4) Tool velocity resulting from individual joint rates.

Figure A2-33 is the VCLR for subroutine POUTC.

Compute Current Tool Position	
Individual Joint Control? (ICNTRL = 2 ?)	
T	F
Null	Compute Ideal Tool Position from Input Rate Profiles
Compute Current Tool Velocity from Current Joint Rates	
Transform Joint-to-Tool Vectors to Base Coordinates	

Figure A2-33 POUTC VCLR

2.2.1.4.3 Write out Motion Calculations (WROUT) - WROUT produces a file to be printed that contains details on the computed motion of the manipulator model. WROUT is only called if the user requests a printed output.

On the first time step only, WROUT lists the initial conditions of the manipulator model. The data output includes:

- 1) For each joint
 - Free axis of rotation,
 - Joint rate profile (if applicable*),
 - Vector describing location of next joint,
 - Initial joint displacement;
- 2) Tool rate profiles (if applicable*);
- 3) Analysis stop time;
- 4) Analysis time increment step size.

On every time step, WROUT lists:

- 1) Time;
- 2) Ideal tool velocity (if applicable*);
- 3) Tool velocity resulting from joint rates;
- 4) Ideal tool position (if applicable*);
- 5) Current tool position;
- 6) Position error as vector and magnitude (if applicable*);
- 7) For each joint
 - Current joint rotation rate,
 - Current joint displacement (degrees and radians),
 - Current joint-to-base transformation matrix,
 - Current joint-to-tool vector (joint and base coordinates),
 - Current location of joint in base coordinates.

*This step is applicable only if coordinated tool rate control was used.

Figure A2-34 is the VCLR for subroutine WROUT.

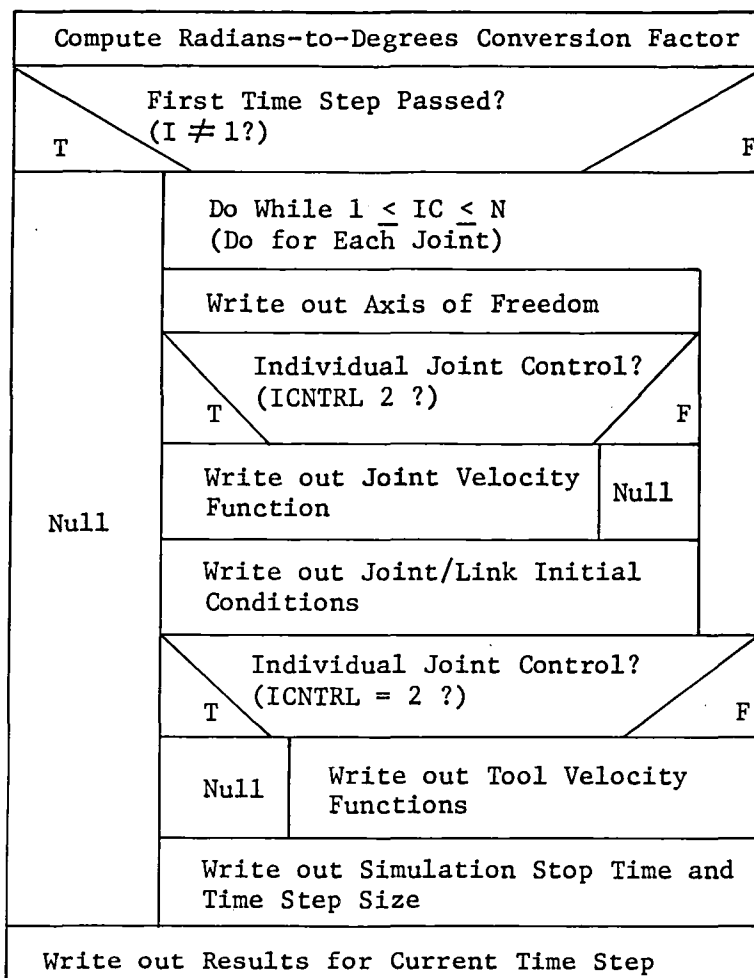


Figure A2-34 WROUT VCLR

2.2.1.4.4 Dynamics Computation Function (DYNAMICS) - Subroutine
DYNAMICS controls the solution of the dynamics equations derived in Appendix B. DYNAMICS is called only if the user has requested dynamics output; in this case, DYNAMICS is called every time step.

For input into the dynamics equations, DYNAMICS calculates the average angular displacement, angular velocity, and angular acceleration for each joint over an entire time step. Average displacement and velocity are obtained by averaging values from the beginning and end of each time step. The average velocities are computed by taking the difference of the velocities from the beginning and end of each time step and dividing by the step size.

DYNAMICS then calls the subroutines that perform the following functions:

- 1) Computation of transformation matrices for the average joint displacements;
- 2) Transformation of all quantities to base coordinates;
- 3) Computation of total velocity and acceleration of each joint/link system;
- 4) Computation of dynamic reactions;
- 5) Writing of results to print file.

Figure A2-35 is the VCLR for subroutine DYNAMICS.

Initialize All Variables to Zero
Average Position and Velocity Over Time Step
Compute Average Acceleration by Differences
Call CCTM: Compute Transformation Matrices for Average Position
Call TQBASE: Transform All Quantities to Base Coordinates
Call CABSM: Compute Absolute Velocities and Accelerations
Call FORCE: Compute Reaction Forces
Call TORQUE: Compute Reaction Torques
Call DYNOUT: Write Results to Print File

Figure A2-35 DYNAMICS VCLR

2.2.1.4.4.1 Transform Quantities to Base Coordinates (TQBASE) - The purpose of subroutine TQBASE is to transform the vector quantities describing manipulator motion and the link inertia matrices into base coordinates.

The formula for transforming vectors from joint i coordinates to base coordinates is

$${}_bV = {}_bT_i {}_iV$$

where

${}_iV$ = Vector, V, expressed in joint i coordinates;
 ${}_bV$ = Same vector V, expressed in base coordinates;
 ${}_bT_i$ = Joint i-to-base transformation matrix.

The formula for transforming inertia matrices from joint coordinates to base coordinates is

$${}_bI = ({}_bT_i)({}_iI)({}_bT_i)^{-1}$$

where

${}_iI$ = Inertia matrix, I, expressed in joint i coordinates;
 ${}_bI$ = Same inertia matrix, I, expressed in base coordinates;
 ${}_bT_i$ = Joint i-to-base transformation matrix.

For a more detailed discussion of the transformation, see Appendix B.

Figure A2-36 is the VCLR for subroutine TQBASE.

Transform Vector Quantities to Base Coordinates
Transform Inertia Matrices to Base Coordinates

Figure A2-36 TQBASE VCLR

2.2.1.4.4.2 Compute Absolute Motion (CABSM) - CABSM uses a recursive technique (see Appendix B) to compute the absolute angular velocity, angular acceleration, and linear acceleration for each joint/link system. CABSM also computes the location of each joint and the tool in base coordinates.

Figure A2-37 is the VCLR for subroutine CABSM.

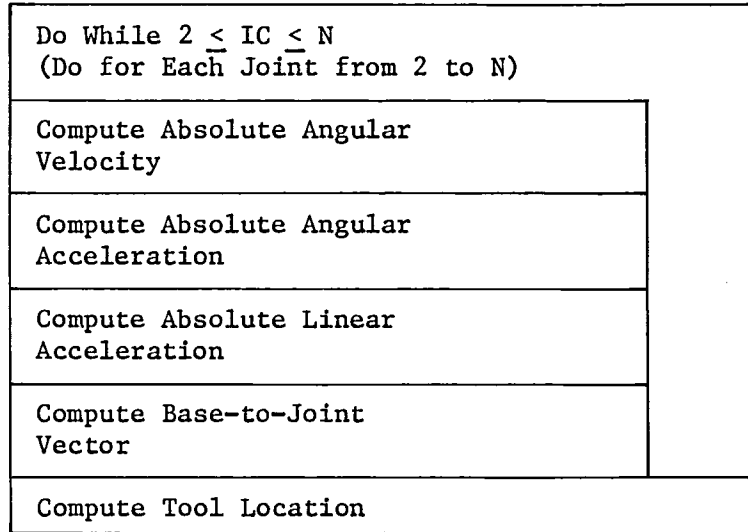


Figure A2-37 CABSM

2.2.1.4.4.3 Compute Reaction Forces (FORCE) - FORCE computes the reaction forces at each joint using the recursive technique derived in Appendix B. In particular, FORCE implements Equation B-12.

FORCE also transforms the joint reaction forces into base coordinates.

Figure A2-38 is the VCLR for subroutine FORCE.

Set Internal Variables	
Do While $N \leq IC \leq 1$ (Do for Each Joint from N Back to 1)	
Compute T4 = Force Due to Centripetal Acceleration	
Compute T3 = Force Due to Angular Acceleration of System	
Compute T2 = Force Due to Linear Acceleration of System (Includes Gravity)	
T1 = Force Transmitted from Joint IC+1	
Force = T1 + T2 + T3 + T4	
Transform Force to Base Coordinates	

Figure A2-38 FORCE VCLR

2.2.1.4.4.4 Compute Reaction Torque (TORQUE) - TORQUE computes the reaction torques at each joint using the recursive technique derived in Appendix B. In particular, TORQUE implements Equation B-13.

TORQUE also transforms the joint reaction forces into base coordinates.

Figure A2-39 is the VCLR for subroutine TORQUE.

Set Internal Variables
Do While $N \geq IC \geq 1$ (Do for Each Joint from N to 1)
$S1 = \text{Torque Transmitted from Joint } IC + 1$
Compute $S2 = I\alpha$
Compute $S3 = w \times Iw$
Compute $S4 = \text{Moment of Force Transmitted from Joint } IC + 1$
$S5 = \text{Any External Torques}$
Compute $S6 = m \{ r \times [w \times (x \times r)] \}$
$\text{Torque} = S1 + S2 + S3 + S4 + S5 + S6$
Transform Torque to Base Coordinates

Figure A2-39 TORQUE VCLR

2.2.1.4.4.5 Write Dynamics Output (DYNOUT) - DYNOUT writes a detailed record of the dynamic input and output to a print file.

In addition to the dynamics output, at the end of the first time step, before any dynamics output has been written, DYNOUT writes a record of the model initial conditions. This initial record includes:

- 1) The number of joints in the manipulator model;
- 2) The analysis start time, stop time, and step size;
- 3) For each joint
 - The joint type,
 - Link dimensions in inches and meters,
 - Location of link center-of-gravity in inches and meters,
 - Mass of link in kilograms,
 - Link inertia matrix.

At the end of every time step, DYNOUT writes a record of the dynamics input and output. For each joint, this record includes:

- 1) Joint angular displacement (degrees and radians);
- 2) Joint location in base coordinates (inches and meters);
- 3) Link vector (inches and meters, i-joint and base coordinates);
- 4) Relative angular velocity in radians/second (joint and base coordinates);
- 5) Relative angular acceleration in radians/second² (joint and base coordinates);
- 6) Absolute angular velocity in radians/second (joint and base coordinates);
- 7) Relative angular acceleration in radians/second² (joint and base coordinates);
- 8) Absolute linear acceleration in inches/second² and meters/second² (joint and base coordinates);
- 9) Reaction force in kilogram-inches/second² and kilogram-meters/second² (joint and base coordinates);
- 10) Reaction torque in kilogram-inches²/second² and kilogram-meters²/second² (joint and base coordinates).

Figure A2-40 is the VCLR for subroutine DYNOUT.

Set Internal Variables	
T	Total Time = Start Time + Step?
Write out Simulation Initial Conditions	Null
Write out Manipulator Physical Parameters	
Write out Values of Position, Velocity, and Acceleration Variables	
Write out Computed Forces and Torques in Joint and Inertial Coordinates	

Figure A2-40 DYNOUT VCLR

2.3 POSTPROCESSING FUNCTION DRIVER (POSTDRVR)

The Postprocessing function driver operates in an interactive mode and prompts the user for the postprocessing option desired. Currently valid options are:

- 1) Replay robotic system motion;
- 2) Parameter versus parameter plots;
- 3) Return to the ROBSIM executive.

Additional postprocessing functions will be added as program capabilities are expanded.

If Option 1 is selected, subroutine MOTION is called to provide a replay of the system motion computed during the Analysis Tools function execution. If Option 2 is selected, subroutine GENPLT is called to provide parameter versus parameter plots of any of the data computed and written to a plot file during the Analysis Tools function execution.

A recoverable error encountered within subroutine POSTDRVR causes an error message to be written through a call to subroutine ERRMSG. Appropriate recovery action is then taken. Nonrecoverable errors encountered within routines called by POSTDRVR cause return of control to the ROBSIM executive program.

Figure A2-41 is the VCLR for subroutine POSTDRVR.

Prompt for Postprocessing Function Mode, IMODE			
DOCASE IMODE			
1	2	3	Def
CALL MOTION to Replay Graphics	CALL GENPLT for Parameter vs Parameter Plots	Return to ROBSIM Executive	Null

Figure A2-41 POSTDRVR VCLR

2.3.1 Replay Motion (MOTION)

Subroutine MOTION is called from subroutine POSTDRV to provide a replay of the robotic system motion produced during the Analysis Tools function execution. The user is prompted for the name of the file containing the data generated during the Analysis Tools function execution. The file is then opened. Subroutine GRAFIX is called to initialize the graphics system and display.

The postprocessing data file is read for each time step to obtain the individual joint displacements. Subroutine GRAFIX is called with the joint displacements at each time step to update the display producing the system motion.

Upon completion of the motion display, the postprocessing data file is closed and subroutine GRAFIX is called to terminate the graphics display.

Figure A2-42 is the VCLR for subroutine MOTION.

CALL RDSIM to Read System Definition File	
Prompt for File Name of Analysis Tool Output file	
Open Analysis Tool Output File	
Initialize Graphics and Display	
	Read Current Joint Variables for System
	CALL GRAFIX to Update Display
DUNTIL All Records Read from File	
Terminate Graphics	
Close Analysis Tool Output File	

Figure A2-42 MOTION VCLR

2.3.2 General Plotting (GENPLT)

Subroutine GENPLT is called from subroutine POSTDRVR to provide parameter versus parameter plots of the data generated during the Analysis Tools function execution. A plot file must have been requested with the Analysis Tools function execution.

Subroutine GENPLT provides the capability to plot the data for any variable on the plot file against any other variable on the plot file. User interface with subroutine GENPLT is through interactive prompts. Plotting may be requested on an HP7221 plotter, Tektronix 4010, Tektronix 4014, or Retrographics terminal. Plotting is done using the DISSPLA plot package.

The user is prompted for the file name of the plot file, which is then opened. Subroutine RDPLT is called to prompt the user for the variables to be plotted. The user may then specify a variety of options or use defaults as desired. The user has complete control of plot format. Once all plot characteristics are specified, subroutine RDPLT is called to read the plot file and extract the data to be plotted.

Following plotting of the data, subroutine LOGO may be called to plot the Martin Marietta logo if desired.

Figure A2-43 is the VCLR for subroutine GENPLT.

Prompt for Plotting Device			
Initialize Requested Plotting Device			
Prompt for File Name of Data File			
Open Data File			
Prompt for Legend Request Flag			
CALL RDPLT to Select Plot Parameters			
Prompt for Plot Symbol Frequency			
Prompt for Data Smoothing Flag			
T	Legend Requested		F
Prompt for Legend Text		Null	
Store Legend Text			
Prompt for Automatic Scaling			
T	Automatic Scaling		F
Null	Prompt for Horizontal Data Max/Min		
	Prompt for Vertical Data Max/Min		
	Prompt for Horizontal and Vertical Tick Increments		
Prompt for Plot Title			
Prompt for Axis Label Option			
T	Specify Axis Label		F
Prompt for Horizontal Axis Label		Use Requested Plot Symbols as Axis Labels	
Prompt for Vertical Axis Label			
Prompt for Page Format Option			

Figure A2-43 GENPLT VCLR

T	Specify Page Format		F	
Prompt for Page Dimensions		Set Parameters for 11.0x8.5 Format		
Prompt for Origin Location		T	Logo Required	
Prompt for Axis Lengths		Set Logo Parameters	Null	
T	Logo Requested			F
Prompt for Logo Position				Null
Prompt for Logo Size				
T	Automatic Scaling		F	
Compute Horizontal Tick Mark Increment			Null	
Compute Horizontal Axis Max/Min Values				
Compute Vertical Tick Mark Increment				
Compute Vertical Axis Max/Min Values				
Draw Axes				
Label Horizontal Axis				
Label Vertical Axis				
Draw Plot Title				
Draw Axes Tick Marks and Label				
CALL RDPLT to Read Plot File and Load Plot Data				
T	Smooth Data		F	
Spline Fit Data		Null		
Plot Data				
T	Legend Required		F	
Plot Legend		Null		
T	Logo Requested		F	
Plot Logo		Null		
Terminate Plotting Device				

Figure A2-43 (concluded)

2.3.2.1 Read Plot File (RDPLT) - Subroutine RDPLT is called from subroutine GENPLT to prompt the user for the variables to be plotted and to read the plot file and extract the data to be plotted.

The input argument IFLAG is used to specify selection of plot parameters or extraction of plot data. If IFLAG = 1, first the plot file symbol record is read. The user is then prompted for the symbol of the parameter desired as the horizontal axis parameter. The symbol record is searched to locate the requested symbol and determine the position of the data corresponding to that symbol in the plot file records. This process is repeated for the vertical axis parameter. The plot file header record is then read to obtain the maximum and minimum values for the plot parameters selected.

If IFLAG = 2, the plot file data records are read to extract and store the parameter values to be plotted. A maximum of 5000 data values can be stored at one time. If more data than that are required, they will be plotted in blocks of 5000 points.

Figure A2-44 is the VCLR for subroutine RDPLT.

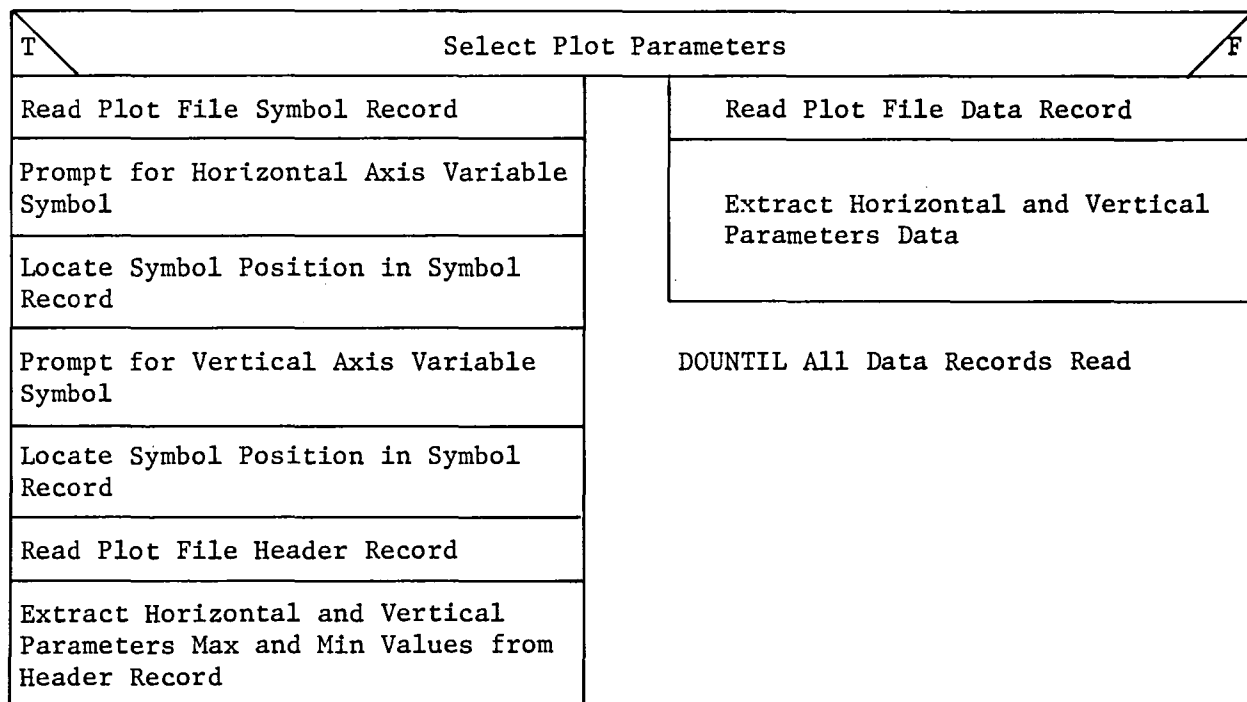


Figure A2-44 RDPLT VCLR

2.3.2.2 Plot Logo (LOGO) - Subroutine LOGO is called from subroutine GENPLT to plot the Martin Marietta logo. The logo can be placed anywhere on the plot and can be drawn any size.

Figure A2-45 is the VCLR for subroutine LOGO.

Open Logo Data File	
Read Logo Data	
Close Logo Data File	
	Draw Logo Segment
DOUNTIL All Logo Segments Complete	

Figure A2-45 LOGO VCLR

2.4 ROBSIM UTILITIES

This section contains the description of ROBSIM subroutines that provide utility functions and are used throughout the program. The routines described in this section are:

- 1) ERRMSG - Searches the error message file and prints the current error message;
- 2) RDSIM - Reads the data file created by the System Definition function;
- 3) SETLU - Sets logical units for all program input and output;
- 4) WRTSIM - Writes to disk the data file generated by the System Definition function;
- 5) ZERCOM - Zeros all locations in the common blocks used for data storage in the System Definition function.

2.4.1 Error Message Routine (ERRMSG)

The subroutine ERRMSG prints error messages for any errors occurring during execution of the ROBSIM program. Error conditions have been defined throughout the program and error messages stored in an error message file with unique error numbers used to identify each error message. When an error occurs, subroutine ERRMSG is called with the appropriate error number. Subroutine ERRMSG opens the error message file and locates and prints the appropriate error message. The error message file is then closed and control is returned to the calling program for appropriate action.

A recoverable error encountered within subroutine ERRMSG causes an error message to be written followed by appropriate recovery action. A nonrecoverable error encountered within subroutine ERRMSG causes an error message to be written and control to be returned through the sequence of calling routines back to the ROBSIM executive program for further user action.

Figure A2-46 is the VCLR for subroutine ERRMSG.

Open Error Message File
Locate Text of Message Corresponding to Input Error Number
Write Error Message
Close Error Message File

Figure A2-46 ERRMSG VCLR

2.4.2 Read Input File (RDSIM)

The subroutine RDSIM reads the data file created by the System Definition function. The data file is an unformatted file containing the contents of the following COMMON blocks:

- 1) COMMON/AMASPR/ - Mass properties data;
- 2) COMMON/CONSTR/ - Constraint data;
- 3) COMMON/CONTRL/ - Prescribed motion data;
- 4) COMMON/ENVIRN/ - Environment graphics representation data;
- 5) COMMON/GEOM/ - System geometry data;
- 6) COMMON/IOBJ/ - System graphics representation data;
- 7) COMMON/IOPT/ - Program options data.

The user is prompted for the file name under which the data have been stored on disk. The file is opened and read, loading all data into the appropriate COMMON locations. The user is then prompted to specify whether the file is to be saved or deleted. The file is then closed with the proper disposition option.

An error encountered within subroutine RDSIM causes an error message to be written through a call to subroutine ERRMSG. If the error is a recoverable error, appropriate recovery action is taken. If the error is nonrecoverable, control is returned through the sequence of calling routines to the ROBSIM executive program for further user action.

Figure A2-47 is the VCLR for subroutine RDSIM.

Prompt for File Name of Input File	
Open Requested File	
Read Mass Properties Data	
Read Constraints Data	
Read Motion Specification Data	
Read Environment Graphics Data	
Read Robotic System Geometry Data	
Read Robotic System Graphics Data	
Read Program Options Data	
Prompt for Disposition of Input File	
T IF Deletion of File Requested F	
Close and Delete File	Close and Save File

Figure A2-47 RDSIM VCLR

2.4.3 Set Logical Units (SETLU)

The subroutine SETLU sets the logical units used within the ROBSIM program for all program input and output. Default logical units are set in a DATA statement in subroutine SETLU. The user may accept the default logical unit assignments or may request modifications. The capability to modify the default logical unit assignments is not currently implemented. The logical unit data are stored in COMMON block/LUNIT/for use by the rest of the ROBSIM program.

Figure A2-48 is the VCLR for subroutine SETLU.

Set Logical Unit Numbers Required for Program
I/O Operations

Figure A2-48 SETLU VCLR

2.4.4 Write Input File (WRTSIM)

The subroutine WRTSIM writes to disk the data file created by the System Definition function. The data file is an unformatted file containing the contents of the following COMMON blocks:

- 1) COMMON/AMASPR/ - Mass properties data;
- 2) COMMON/CONSTR/ - Constraint data;
- 3) COMMON/CONTRL/ - Prescribed motion data;
- 4) COMMON/ENVIRN/ - Environment graphics representation data;
- 5) COMMON/GEOM/ - System geometry data;
- 6) COMMON/IOBJ/ - System graphics representation data;
- 7) COMMON/IOPT/ - Program options data.

The user is prompted for the file name under which the data are to be stored on disk. The file is opened and the data from each COMMON block are written to the file. The file is then closed with a disposition option of "SAVE."

An error encountered within subroutine WRTSIM causes an error message to be written through a call to subroutine ERRMSG. If the error is a recoverable error, appropriate recovery action is taken. If the error is nonrecoverable, control is returned through the sequence of calling routines to the ROBSIM executive program for further user action.

Figure A2-49 is the VCLR for subroutine WRTSIM.

Prompt for File Name To Be Used
Open Requested File
Write Mass Properties Data from Common /AMASPR/ to Disk File
Write Constraints Data from Common /CONSTR/ to Disk File
Write Motion Specification Data from Common /CONTRL/ to Disk File
Write Environment Graphics Data from Common /ENVIRN/ to Disk File
Write Robotic System Geometry Data from Common /GEOM/ to Disk File
Write Robotic System Graphics Data from Common /IOBJ/ to Disk File
Write Program Options Data from Common /IOPT/ to Disk File
Close File

Figure A2-49 WRTSIM VCLR

2.4.5 Zero COMMON (ZERCOM)

The subroutine ZERCOM zeros COMMON locations used to store the data that are input during the System Definition function. Subroutine ZERCOM is called from subroutine CREATE prior to defining a new data file and zeros the contents of the following COMMON blocks:

- 1) COMMON/AMASPR/ - Mass properties data;
- 2) COMMON/CONSTR/ - Constraint data;
- 3) COMMON/CONTRL/ - Prescribed motion data;
- 4) COMMON/ENVIRN/ - Environment graphics representation data;
- 5) COMMON/GEOM/ - System geometry data;
- 6) COMMON/IOBJ/ - System graphics representation data;
- 7) COMMON/IOPT/ - Program options data.

Figure A2-50 is the VCLR for subroutine ZERCOM.

Zero Fill Mass Properties Common /AMASPR/
Zero Fill Constraints Common /CONSTR/
Zero Fill Motion Specification Common /CONTRL/
Zero Fill Environment Graphics Data Common /ENVIRN/
Zero Fill Robotic System Geometry Common /GEOM/
Zero Fill Robotic System Graphics Data Common /IOBJ/
Zero Fill Program Options Common /IOPT/

Figure A2-50 ZERCOM VCLR

2.5 MATH UTILITIES

This section describes several matrix math routines used within the ROBSIM program.

2.5.1 Compute Complete Transformation Matrix (CCTM)

CCTM is a utility routine that computes the complete transformation matrices needed to transform vectors in the coordinate system of joint i into terms of either the joint $i-1$ or the base coordinate systems.

CCTM is called at each time step. CCTM multiplies the transformation matrices resulting from initial orientation and current joint displacement to obtain the transformation matrix between joint i and joint $i-1$. The joint-to-joint transformation matrices are then multiplied to yield the joint-to-base transformation matrices.

Figure A2-51 is the VCLR for subroutine CCTM.

Compute Degrees-to-Radians Conversion Factor
Set All Transformation Matrices to Identity
Do While $1 \leq IC \leq N$ (Do for Each Joint from Base to Tool)
Set Scratch Matrix to Identity
Compute Transformation Matrix to Joint IC-1 Coordinates Resulting from Initial Orientation
Compute Transformation Matrix from Current Displacement of Joint IC to Initial Displacement
Multiply Matrices to Obtain Complete Joint IC to IC-1 Transformation Matrix
Multiply Matrices to Obtain Complete Joint IC to Base Transformation Matrix

Figure A2-51 CCTM VCLR

2.5.2 Solve Linear System (SLVLIN)

Subroutine SLVLIN is a routine designed to solve the matrix equation $AX = B$ by computing A^+ , the psuedo-inverse of A. Details of the calculation of A^+ , as well as a general discussion of the pseudo-inverse can be found in Appendix D.

Figure A2-52 is the VCLR for subroutine SLVLIN.

Set Internal Variables $Ax = B$
$IMAX = \text{MAX} \{ \text{No. of Rows, No. of Columns} \}$
Call GAUSS: Reduce Augmented System to Row-echelon Form
Compute C in $A = C \cdot D$, a Rank Factorization
Compute Pseudo - Inverse $A^+ = D^t (DD^t)^{-1} C^t$
Compute Best Solution $\hat{x} = A^+ B$

Figure A2-52 SLVLIN VCLR

Perform Gaussian Elimination (GAUSS) - GAUSS is a routine that performs Gauss-Jordan elimination with partial pivoting on an augmented matrix system to reduce the system to row-echelon form. More on the Gauss-Jordan method, pivoting, and row-echelon forms can be found in Appendix D and in Matrices and Linear Transformation.*

Figure A2-53 is the VCLR for subroutine GAUSS.

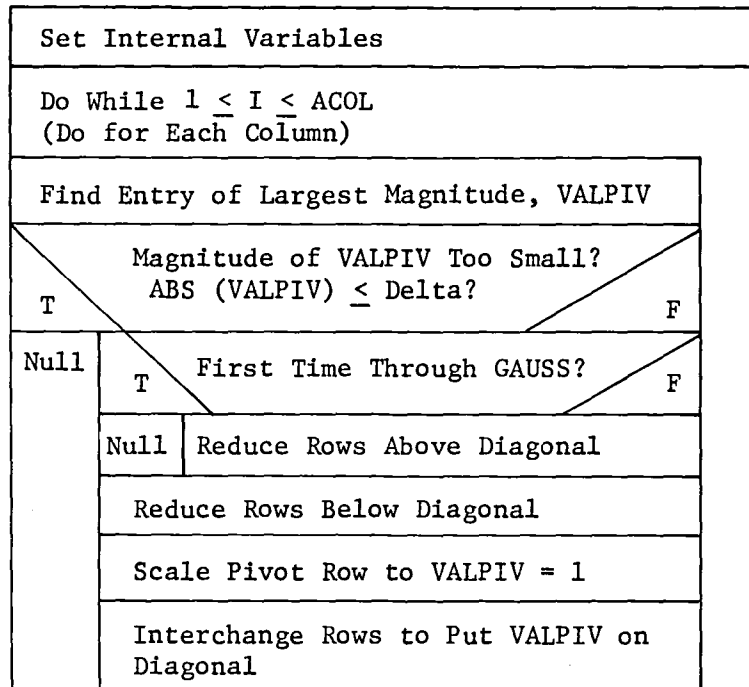


Figure A2-53 GAUSS VCLR

*Charles G. Cullen: Matrices and Linear Transformations. Addison-Wesley Publishing Company, 1972

2.5.3 Matrix Multiplication Function (MATMPY)

MATMPY is a utility routine that performs matrix multiplication. The maximum dimension of any of the matrices is 10x10. When MATMPY is called, the maximum dimension of the calling arguments must be passed to MATMPY.

Figure A2-54 is the VCLR for subroutine MATMPY.

Compute	$C = A \cdot B$
Where	A is $i \times j$
	B is $j \times k$
and	
	C is $i \times k$

Figure A3-54 MATMPY VCLR

2.5.4 Form Skew-Symmetric Matrix (SKEW)

SKEW is a utility routine used to form a 3x3 skew-symmetric matrix given a 3x1 vector as input. The resulting skew-symmetric matrix can be used to evaluate a vector cross product by using matrix multiplication.

Figure A2-55 is the VCLR for subroutine SKEW.

Set Diagonal to Zero
Form Skew-Symmetric Matrix for Cross-Product Operator

Figure A2-55 SKEW VCLR

2.5.5 Compute Elementary Transformation Matrix (CETM)

CETM is a utility routine that is used to calculate the elementary transformation matrix resulting from a rotation about a single coordinate axis. If ϕ is the angle of rotation and T is the resulting transformation matrix, then

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad \text{if } \phi \text{ is a rotation about the x axis;}$$

$$= \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{bmatrix} \quad \text{if } \phi \text{ is a rotation about the y axis;}$$

$$= \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{if } \phi \text{ is a rotation about the z axis;}$$

Figure A2-56 is the VCLR for subroutine CETM.

Set Matrix to Zero		
Do Case (IAXIS - 2)		
IAXIS - 2 < 0	IAXIS - 2 = 0	IAXIS - 2 > 0
Compute Transformation Matrix for Rotation about the X - Axis	About the Y - Axis	About the Z - Axis

Figure A2-56 CETM VCLR

2.6 EVANS AND SUTHERLAND GRAPHICS ROUTINES

The computer-generated motion representations in the ROBSIM program are produced on Evans and Sutherland (E&S) Multi-Picture System (MPS) line drawing graphics hardware. Picture System 2 (PS-2) equipment and MPS FORTRAN callable graphics routines are used. All graphics programming has been confined to a small number of ROBSIM routines to facilitate conversion to another graphics package if required. The new E&S Picture System 300 (PS-300) would be well suited for ROBSIM use. Conversion of the current MPS code would be required to execute ROBSIM on the PS-300 system.

This section briefly describes each of the E&S MPS routines used in ROBSIM. Complete descriptions can be found in the Evans and Sutherland Multi-Picture System Users Manual (document number E&S #901141-052 NC), Chapter 6, "Multi-Picture System Graphics Software Package."

2.6.1 Initialization and Setup

- 1) CALL LSPEED - Set line generator refresh speed;
- 2) CALL MPINIT - Attach MPS Picture Station to user task and initialize;
- 3) CALL SINIT - Clear refresh display file and initialize segment namestack;
- 4) CALL TINIT - Initialize transformation stack and set Picture Processor transformation to identify;
- 5) CALL TWINDP - Set three-dimensional perspective window;
- 6) CALL VBOUND - Set viewport boundaries;
- 7) CALL VINTEN - Set intensity levels.

2.6.2 Segment Control

- 1) CALL SADD - Add segment to refresh display file;
- 2) CALL SCLOSE - Close most recently opened segment;
- 3) CALL SMARK - Mark specified segment for update;
- 4) CALL SOPEN - Open specified segment;
- 5) CALL SREMOV - Remove specified segment from refresh display file;
- 6) CALL SREP - Replace the marked segment in the refresh display file with the set of outstanding segments.

2.6.3 Matrix Manipulation

- 1) CALL TCON - Concatenate the specified matrix with the current Picture Processor transformation;
- 2) CALL TGET - Load into the specified matrix the current Picture Processor transformation;
- 3) CALL TPOP - Pop the top element of the Picture Processor transformation stack making it the Picture Processor transformation;
- 4) CALL TPUSH - Push the current Picture Processor transformation onto the transformation stack;
- 5) CALL TROT X - Concatenate with the Picture Processor transformation an x-axis rotation matrix for the specified angle;
- 6) CALL TROT Y - Concatenate with the Picture Processor transformation a y-axis rotation matrix for the specified angle;
- 7) CALL TROT Z - Concatenate with the Picture Processor transformation a z-axis rotation matrix for the specified angle;
- 8) CALL TSCALE - Concatenate with the Picture Processor transformation a scaling matrix for the specified scale values;
- 9) CALL TSET - Set the Picture Processor transformation to the specified matrix;
- 10) CALL TTRAN - Concatenate with the Picture Processor transformation a translation matrix for the specified translation values.

2.6.4 Line Drawing

- 1) CALL D3DATA - Process three-dimensional data array according to line drawing mode selected;
- 2) CALL LCOLOR - Set line-generator color and saturation values.

2.6.5 Termination

CALL MPSTOP - Detach the MPS Picture Station from the user task.

2.7 DISSPLA PLOT ROUTINES

The requirement for x-y type parameter plots is met in the ROBSIM program by the use of the Integrated Software Systems Corporation (ISSCO) DISSPLA plotting package. Conversion to another plot package could be easily accomplished.

This section provides a brief description of the DISSPLA routines used in ROBSIM. Complete descriptions of these routines can be found in the Display Integrated Software System and Plotting Language (DISSPLA) Users Manual, Version 9.0.

2.7.1 Device Initialization

- 1) CALL HP7221 - Initialize Hewlett Packard HP7221 plotter;
- 2) CALL TK4010 - Initialize Tektronix 4010 terminal or Retrographics terminal;
- 3) CALL TK4014 - Initialize Tektronix 4014 terminal.

2.7.2 Plot Layout

- 1) CALL AREA2D - Define the subplot area based on input axis lengths;
- 2) CALL GRAF - Primary graph setup routine that establishes relationship between physical axis length and plot units and draws axes;
- 3) CALL HEADIN - Write plot heading (title);
- 4) CALL NOBRDR - Suppress drawing of border around graph layout;
- 5) CALL PAGE - Set page size in inches;
- 6) CALL PHYSOR - Define location of origin on page;
- 7) CALL SETCLR - Set pen color;
- 8) CALL XNAME - Label x axis;
- 9) CALL YNAME - Label y axis.

2.7.3 Curve Plotting

- 1) CALL CURVE - Plot input x and y data;
- 2) CALL LINEAR - Plot data linearly connected point to point;
- 3) CALL RASPLN - Smooth data with rational spline fit.

2.7.4 Point-to-Point Plotting

- 1) CALL CONNPT - Draw line from current location to input location;
- 2) CALL STRTPT - Move to input location.

2.7.5 Legend

- 1) CALL LEGEND - Draw legend;
- 2) CALL LINES - Load legend text;
- 3) Function LINEST - Set line length for legend text.

2.7.6 Termination

- 1) CALL DONEPL - Terminate plot device;
- 2) CALL ENDPL - Terminate current plot.

3.0 JNTMOD - MAIN PROGRAM

The basis of what will become the simulation tool within the ROBSIM program is being developed outside the ROBSIM framework as the Joint Model (JNTMOD) program. The JNTMOD program models a single typical joint using the state-variable formulation. The single joint model allows demonstration of the state-variable concept and will be extended first to a two-link case and ultimately to an N-link capability. The joint model consists of controller, amplifier, motor, power train, and load models. A Kalman filter is also provided.

The basic state-variable formulation equations used in each model block are:

$$X(k+1) = \phi(k+1,k)X(k) + \theta(k+1,k)U(k) + W(k)$$

$$Y(k+1) = C(k+1)X(k+1)$$

$$Z(k+1) = H(k+1)Y(k+1) + V(k)$$

where

U = Control function;

X = State-variable array;

W = Process noise;

Y = Observable array;

Z = Sensor output array;

V = Sensor noise.

Figure A3-1 shows a typical model block.

When the individual model blocks are combined to form the joint model, a system of equations in the state-variable formulation is formed. Figure A3-2 shows the joint model configuration. A complete discussion of the simulation tool formulation is given in Subsection IV.C of the ROBSIM final report. This document describes the individual subroutines that make up the JNTMOD program. It assumes that the reader is familiar with the notation and equations used.

The Joint Model program main routine, JNTMOD, controls the logic flow through the program. Subroutine INPUT is called to provide input of the parameters necessary within each of the model blocks. Most program options are input through subroutine INPUT as well. Subroutine INIT provides initialization for the joint model system. Subroutine KFINIT is called to handle initialization required if the Kalman filter has been requested. The remainder of the JNTMOD code is within a time loop from the user-requested simulation start time to the user-requested simulation stop time.

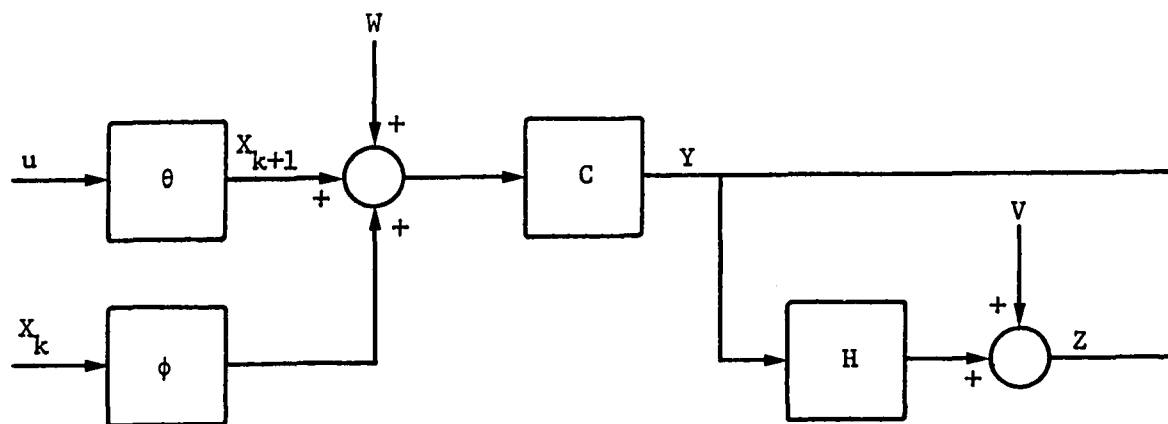


Figure A3-1 Typical Model Block

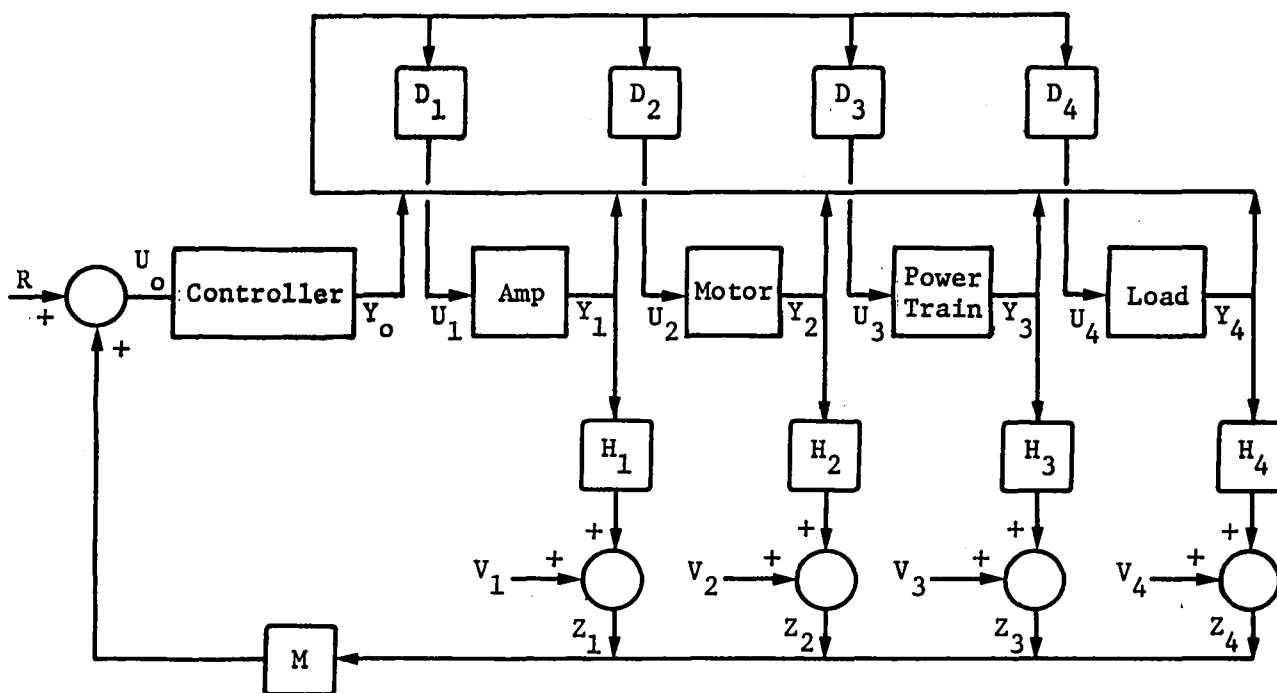


Figure A3-2 Joint Model

The controller is modeled in subroutine CONTRL using the state-variable formulation. If plot file output is requested, the controller parameters are loaded in the plot file data record through calls to subroutine LDVALU.

The logic sequence for each of the remaining joint model blocks (amplifier, motor, power train, and load) is identical. Subroutine UVEC is called to load the control array. The appropriate block model subroutine is then called (AMP, MOTOR, PWRTRN, or LOAD). Each block model uses the state-variable formulation. If plot file output is requested, subroutine LDVALU is called to load the block model parameters in the plot file data record. If printed output is requested, subroutine PRT is called.

If the Kalman filter is requested, the system sensor output array is loaded in subroutine BLDZAL. The system phi matrix is constructed in subroutine BLDPHI. The system theta matrix is constructed in subroutine BLDTHT. The system sensor transform is constructed in subroutine BUILDH. Subroutine KALMAN is called to perform the Kalman filter calculations. If plot file output is requested, the Kalman filter parameters are loaded in the plot file data record through calls to subroutine LDVALU.

The results of the Kalman filter calculations may be used in the system feedback loop if requested.

Following completion of the time loop, the plot file header record is written, if required, and all open files are closed.

Figure A3-3 is the VCLR for the JNTMOD routine.

CALL INPUT to Obtain User Input Data		
Set Logical Unit Numbers		
Prompt for Kalman Filter Flag, KFFLAG		
T	KFFLAG.EQ.1	F
Prompt for Use Filter Results in Feed-back Loop Flag, KFFDBK		Null
T	Printed Output Requested	F
Open Print File		Null
T	Plotted Output Requested	F
Open Plot File		Null
CALL INIT to Initialize All Matrices and State, Observable, and Sensor Output Arrays		
T	KFFLAG.EQ.1	F
CALL KFINIT to Initialize All Matrices and State, Observable, and Sensor Output Arrays for Kalman Filter Computations		Null
TIME = Start Time		
TIME = TIME + Step Size		
CALL CONTRL to Load the Control Array		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
CALL UVEC to Load the Amplifier Control Array		
CALL AMP (Amplifier Model)		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
T	Printed Output Requested	F
CALL PRT to Print Amplifier Model Output		Null
CALL UVEC to Load the Motor Control Array		
CALL MOTOR (Motor Model)		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
T	Printed Output Requested	F
CALL PRT to Print Motor Model Output		Null

Figure A3-3 JNTMOD VCLR

CALL UVEC to Load the Power Train Control Array		
CALL PWRTRN (Power Train Model)		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
T	Printed Output Requested	F
CALL PRT to Print Power Train Model Output		Null
CALL UVEC to Load the Load Control Array		
CALL LOAD (Load Model)		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
T	Printed Output Requested	F
CALL PRT to Print Load Model Output		Null
T	KFFLAG.EQ.1	F
CALL BLDZAL to Build System Sensor Output Array and System Control Array		Null
CALL BLDPHI to Build System PHI Matrix		
CALL BLDTHI to Build System THETA Matrix		
CALL BUILDH to Build System H Matrix		
CALL KALMAN (Kalman Filter Model)		
T	Plotted Output Requested	F
CALL LDVALU to Load Plot File Array		Null
T	KFFDBK.EQ.1	F
Replace System Sensor Output with Kalman Filter Output		Null
T	Plotted Output Requested	F
Output Record to Plot File		Null
DOUNTIL [(Stop Time - TIME) .LE. Step Size/2]		
T	Plotted Output Requested	F
CALL WRTHDR to Write Plot File Header Record		Null
Close Plot File		
T	Printed Output Requested	F
Close Print File		Null

3.1 GENERAL JOINT MODEL ROUTINES

This section contains descriptions of routines used for input, initialization, and printing. Also described are routines used in each of the model blocks for loading the control array and computing noise, and for sensor computations.

3.1.1 Program Input (INPUT)

Subroutine INPUT is called from the program main routine JNTMOD to handle the bulk of program inputs. Input may be totally through program prompts or a previously created disk file may be read to provide most of the data describing the system to be modeled.

If manual input of the joint model data is requested, the user is first prompted for controller data. Dimensions are required for the controller control array, state-variable array, and observable array. The constants or coefficients required for the controller in use are then input. The user then selects open- or closed-loop control. Finally, the user must specify the standard deviations to be used in noise calculations for each of the controller state variables.

Following input of the controller data, the user must input data for each of the joint component models (amplifier, motor, power train, and load). The same data are required for each of the model blocks. First are the dimensions of the control, state-variable, observable, and sensor arrays. The constants or coefficients required for the model in use are then input. Finally, the standard deviation for each state variable and each sensor output variable are input for use in noise calculations.

The user may elect to write to disk file the data input defining the current joint model. This disk file can then be used as input for future executions.

Regardless of whether the previous data were introduced manually or read from disk file, the user is provided the opportunity to modify selected data parameters. The user may modify any of the constants, coefficients, or standard deviations for the controller, amplifier, motor, power train, or load models.

The user is then prompted for a seed value for random number generation; the simulation start time, stop time, and time step; and print and plot output flags.

Figure A3-4 is the VCLR for subroutine INPUT.

Set Logical Unit Numbers	
Prompt for Input Mechanism-Manual or via Existing Data File	
T	Manual Input
Prompt for Control Model Data: <ul style="list-style-type: none"> . Dimensions for Input, State Variable, and Output Arrays . Control Model Constants . Selection of Open or Closed-Loop Control . Standard Deviation Values for Control Model State Noise 	Prompt for File Name
Prompt for Amplifier Model Data: <ul style="list-style-type: none"> . Dimensions for Input, State Variables, Output, and Sensor Output Arrays . Amplifier Model Constants . Standard Deviation Values for Amplifier Model State Noise . Standard Deviation Values for Amplifier Model Sensor Noise 	Read Input File
Prompt for Motor Model Data: <ul style="list-style-type: none"> . Dimensions for Input, State Variables, Output and Sensor Output Arrays . Motor Model Constants . Standard Deviation Values for Motor Model Noise . Standard Deviation Values for Motor Model Sensor Noise 	

Figure A3-4 INPUT VCLR

Prompt for Power Train Model Data: <ul style="list-style-type: none"> . Dimensions for Input, State Variable, Output, and Sensor Output Arrays . Power Train Model Constants . Standard Deviation Values for Power Train Model State Noise . Standard Deviation Values for Power Train Model Sensor Noise 		Read Input File
Prompt for Load Model Data: <ul style="list-style-type: none"> . Dimensions for Input, State Variable, Output, and Sensor Output Arrays . Load Model Constants . Standard Deviation Values for Load Model State Noise . Standard Deviation Values for Load Model Sensor Noise 		
Prompt for Input Save Option		
T	Save Input to Disk File Option Selected	
Prompt for File name		Null
Write Input Data to File		
Prompt for Modifications to Control Model Data <ul style="list-style-type: none"> . Control Constants . Standard Deviations for Control Model State Noise 		
Prompt for Modifications to Amplifier Model Data: <ul style="list-style-type: none"> . Standard Deviations for Amplifier Model State Noise . Standard Deviations for Amplifier Model Sensor Noise 		

Figure A3-4 (Continued)

<p>Prompt for Modifications to Motor Model Data:</p> <ul style="list-style-type: none"> . Standard Deviations for Motor Model State Noise . Standard Deviations for Motor Model Sensor Noise
<p>Prompt for Modifications to Power Train Model Data:</p> <ul style="list-style-type: none"> . Standard Deviations for Power Train Model State Noise . Standard Deviations for Power Train Model Sensor Noise
<p>Prompt for Modifications to Load Model Data:</p> <ul style="list-style-type: none"> . Standard Deviations for Load Model State Noise . Standard Deviations for Load Model Sensor Noise
Prompt for Random Number Generator Seed Value
Prompt for Start Time, Stop Time, and Time Stop
Prompt for Print Output Frequency
Prompt for Plot Output Frequency

Figure A3-4 (Concluded)

3.1.2 Joint Model Initialization (INIT)

Subroutine INIT is called from the main routine JNTMOD to handle initialization of the joint model arrays and matrices. The state-variable, observable, and sensor arrays for each of the joint model blocks are set to initial values. All joint model block phi, theta, C, and M matrices are computed for the simulation start time. The D matrices used to develop the control arrays, and the feedback matrix, M, are set. The reference signal array, R, is loaded with start time values. Finally, if printed output is requested, subroutine PTINIT is called to write the system initial conditions.

Figure A3-5 is the VCLR for subroutine INIT.

Zero the State Variable Arrays	
Load State Variable Array Elements with Initial Values	
Zero the Observable Arrays	
Zero the Sensor Output Arrays	
Initialize All Matrices at Start Time	
T	Printed Output Requested F
CALL PTINIT to Print Initial Conditions	Null

Figure A3-5 INIT VCLR

Print Initial Conditions (PTINIT) - Subroutine PTINIT is called from subroutine INIT to print the joint model initial conditions. First, the requested simulation start time, stop time, and time step are printed. Then the initial state-variable, observable, and sensor arrays for the amplifier, motor, power train, and load are printed.

Figure A3-6 is the VCLR for subroutine PTINIT.

Start New Page and Print Page Header
Print Start Time, Stop Time, and Time Step
Print Amplifier Initial Conditions: <ul style="list-style-type: none"> . State Array . Observable Array . Sensor Array
Print Motor Initial Conditions: <ul style="list-style-type: none"> . State Array . Observable Array . Sensor Array
Print Power Train Initial Conditions: <ul style="list-style-type: none"> . State Array . Observable Array . Sensor Array
Print Load Initial Conditions: <ul style="list-style-type: none"> . State Array . Observable Array . Sensor Array

Figure A3-6 PTINIT VCLR

3.1.3 Load Control Array (UVEC)

Subroutine UVEC is called from the main routine JNTMOD prior to calls to each of the joint model component routines (AMP, MOTOR, PWRTRN, and LOAD). Subroutine UVEC loads the control array, U, required for each model block. The control array is composed of observable parameters from any of the joint model blocks (controller, amplifier, motor, power train, or load). The system observable array, CAPY, is loaded with the elements of the individual joint model block observable arrays. Subroutine COMPD is called to compute the D matrix for the requested model block. The D matrix provides the relationship between the system observable array and the control array for the requested model block. Subroutine MATMPY is called to multiply the D matrix and the CAPY array to produce the required control array, U.

Figure A3-7 is the VCLR for subroutine UVEC.

Zero the Control Array, U	
Load the Total Y Array, CAPY, with:	
<ul style="list-style-type: none"> . Observable Array from the Control Box . Observable Array from the Amplifier . Observable Array from the Motor . Observable Array from the Power Train . Observable Array from the Load 	
T	F
D Matrix a Function of Time	
CALL COMPD to Compute the Matrix, D, Used to Relate the Combined Observable Output Array to the Input Array	Null
CALL MATMPY to Multiply CAPY by D and Produce the Control Array, U	

Figure A3-7 UVEC VCLR

Compute D Matrix (COMPD) - Subroutine COMPD is called from subroutine UVEC to compute the D matrix for the requested joint model block (amplifier, motor, power train, or load). The control array required as input to each of the model blocks is composed of elements of the observable arrays from throughout the system. The D matrix relates the system observables to the control array for the requested model block.

Figure A3-8 is the VCLR for subroutine COMPD.

Zero the Matrix, D, Used to Relate Combined Observable Output Array to the Input Array				
DOCASE IFLAG				
1	2	3	4	Def
Compute D Matrix for Amplifier	Compute D Matrix for Motor	Compute D Matrix for Power Train	Compute D Matrix for Load	Null

Figure A3-8 COMPD VCLR

3.1.4 Process Noise (XNOISE)

Subroutine XNOISE is called from each of the model subroutines (AMP, MOTOR, PWRTRN, and LOAD) to compute the state-variable noise. The noise array, W, is first zeroed. Subroutine GGNML of the IMSL math package is called to compute a normal random value from a Gaussian distribution with zero mean and variance of one. The state noise, W, is then computed as the user-specified standard deviation for state noise times the value obtained from subroutine GGNML.

Figure A3-9 is the VCLR for subroutine XNOISE.

Zero the Noise Array
CALL GGNML to Obtain a Sample from a Gaussian Distribution with Zero Mean and Variance of One
Compute Noise Array for the State Variable Array (User-Specified Standard Deviation Times Sample Value)

Figure A3-9 XNOISE VCLR

3.1.5 Compute C Matrix (COMPC)

Subroutine COMPC is called from each model block routine (AMP, MOTOR, PWRTRN, or LOAD) to compute the C matrix for the requested model block. The C matrix transforms the state-variable array to the observable array in the state-variable formulation.

The C matrix is first zeroed. The nonzero matrix elements for the requested block are then set.

Figure A3-10 is the VCLR for subroutine COMPC.

Zero Fill C Matrix				
DOCASE IFLAG				
1	2	3	4	Default
Compute C Matrix Elements for Amplifier	Compute C Matrix Elements for Motor	Compute C Matrix Elements for Power Train	Compute C Matrix Elements for Load	Null

Figure A3-10 COMPC VCLR

3.1.6 Sensor Model (SENSOR)

Subroutine SENSOR is called from each of the model subroutines (AMP, MOTOR, PWRTN, or LOAD) to compute the sensor output. Subroutine SENSOR uses the state-variable formulation for sensor output computations. Subroutine HCOMP is called to provide the sensor transform matrix, H, for the specified model. Subroutine MATMPY is called to multiply the H matrix with the C matrix for the specified model to produce the CAPH matrix. The CAPH matrix transforms the state-variable array, X, to the sensor output array without noise, ZWON. Subroutine ZNOISE is called to compute the noise array, V, for the specified model sensor output. The final sensor output array, Z, is produced by adding the sensor noise array, V, to the sensor output without noise array, ZWON.

Figure A3-11 is the VCLR for subroutine SENSOR.

T	H Matrix a Function of Time	F
CALL HCOMP to Compute the Matrix, H, Used to Transform Observable Array to Sensor Output Array		Null
T	H Matrix or C Matrix a Function of Time	F
CALL MATMPY to Multiply C Matrix (Used to Transform State Variable Array to Observable Array) by H Matrix and Produce Matrix CAPH		Null
CALL MATMPY to Multiply State Variable Array, X, by CAPH to Produce Sensor Output without Noise, ZWON		
CALL ZNOISE to Compute the Sensor Noise Array, V		
CALL MATADD to Add V to ZWON and Produce the Final Sensor Output Array, Z		

Figure A3-11 SENSOR VCLR

3.1.6.1 Compute H Matrix (HCOMP) - Subroutine HCOMP is called from subroutine SENSOR to compute the H matrix for the specified joint model component (amplifier, motor, power train, or load). The H matrix is used to transform the observable array into the sensor output array in the state-variable formulation. The H matrix is first zeroed. The nonzero elements for the specified model component are then set.

Figure A3-12 is the VCLR for subroutine HCOMP.

Zero the Matrix, H, Used to Transform the Observable Array to the Sensor Output Array				
DOCASE IFLAG				
1	2	3	4	Def
Compute H Matrix for Amplifier	Compute H Matrix for Motor	Compute H Matrix for Power Train	Compute H Matrix for Load	Null

Figure A3-12 HCOMP VCLR

3.1.6.2 Sensor Noise (ZNOISE) - Subroutine ZNOISE is called from subroutine SENSOR, which is called from each of the model subroutines (AMP, MOTOR, PWRTRN, and LOAD). Subroutine ZNOISE computes the sensor noise. The noise array, V, is first zeroed. Subroutine GGNML of the IMSL math package is called to compute a normal random value from a Gaussian distribution with zero mean and variance of one. The sensor noise, V, is then computed as the user-specified standard deviation for sensor noise times the value obtained from subroutine GGNML.

Figure A3-13 is the VCLR for subroutine ZNOISE.

Zero the Noise Array
CALL GGNML to Obtain a Sample from a Gaussian Distribution with Zero Mean and Variance of One
Compute Noise Array for Sensor Output (User-Specified Standard Deviation Times Sample Value)

Figure A3-13 ZNOISE VCLR

3.1.7 Print Joint Model Parameters (PRT) - Subroutine PRT is called from the main routine JNTMOD following the call to each of the system model routines (AMP, MOTOR, PWRTRN, or LOAD). Subroutine PRT prints all of the parameters associated with that model. Each call to subroutine PRT results in a block of output that contains the following data:

- 1) Current time;
- 2) State-variable arrays--X, XHOM, XPAR, XWON, and W;
- 3) Observable array--Y;
- 4) Sensor arrays--Z, ZWON, and V.

Figure A3-14 is the VCLR for subroutine PRT.

T	New Time Step	F
Skip to New Page and Print Page Header with Current Time		Null
Print Control Array		
Print State Variable Arrays Including X Total, X Homogeneous, X Particular, X without Noise, and X Noise		
Print Observable Array		
Print Sensor Output Arrays Including Z with Noise, Z without Noise, and Z Noise		

Figure A3-14 PRT VCLR

3.2 CONTROLLER MODEL (CONTRL)

Subroutine CONTRL models the controller using the state-variable formulation. Subroutine COMPM is called to compute the feedback matrix, M. The sensor output from all model blocks is combined into a system sensor array, CAPZ. If closed-loop control is requested, subroutine MATMPY is called to multiply the system sensor array with the feedback matrix to produce the feedback array. If open-loop control is requested, the feedback array is set to zero. Subroutine LOADR is then called to load the reference input array, R. The ALPHA array used as input to the Kalman filter routine is loaded with the elements of the R and CAPZ arrays. The joint model control array, U, is formed by adding the feedback array with the reference input array through a call to subroutine MATADD.

Subroutine PHIC is called to compute the state-variable formulation phi matrix for the controller. Subroutine THETAC is called to compute the state-variable formulation theta matrix for the controller. Subroutine MATMPY is called to multiply the phi matrix and the current controller state-variable array, X, to produce the homogeneous state array, XHOM. Subroutine MATMPY is then called to multiply the theta matrix with the control matrix, U, to produce the particular state array, XPAR. The state array without noise is the sum of XHOM and XPAP. Subroutine CNOISE is called to compute the controller process noise array, W. The process noise, W, plus XWON produces the updated controller state variable array, X.

Subroutine CONTC is called to compute the controller state-to-observable transform matrix, C. Subroutine MATMPY is called to multiply the C matrix and the state array, X, to produce the controller observable array, Y.

Figure A3-15 is the VCLR for subroutine CONTRL.

T	M Matrix a Function of Time	F
CALL COMPM to Compute the Matrix, M, Used to Relate Sensor Output to Reference Input Array		Null
Load the System Sensor Output Array, CAPZ, with Sensor Output Arrays from the Amplifier, Motor, Power Train, and Load		
T	.NOT. Open Loop	F
CALL MATMPY to Multiply CAPZ by M and Obtain Feedback Array, U		Zero the Feedback Array, U
T	R Matrix a Function of Time	F
CALL LOADR to Load the Reference Input Array, R		Null
Load R and CAPZ Arrays into ALPHA Array Used as Input to Kalman Filter		
CALL MATADD to Add R to U and Produce the Total Control Array, U		
T	Controller PHI Matrix a Function of Time	F
CALL PHIC to Compute Controller PHI Matrix		Null
T	Controller THETA Matrix a Function of Time	F
CALL THETAC to Compute Controller THETA Matrix		Null
CALL MATMPY to Multiply State Variable Array, X, by PHI to Produce X Homogeneous, XHOM		
CALL MATMPY to Multiply U by THETA and Produce X Particular, XPAR		
CALL MATADD to Add XHOM to XPAN and Produce X without Noise, XWON		
CALL CNOISE to Compute Control Noise Array, W		
CALL MATADD to Add W to XWON and Produce the Final State Variable Array, X		
T	C Matrix a Function of Time	F
CALL CONTC to Compute Matrix, C, Used to Transform X to the Observable Array Y		Null
CALL MATMPY to Multiply X by C and Produce the Observable Array Y		

Figure A3-15 CONTRL VCLR

3.2.1 Compute Feedback Matrix (COMPM)

Subroutine COMPM is called from subroutine CONTRL to compute the feedback matrix, M. The M matrix relates the sensor output from all joint model blocks to the reference input for feedback purposes.

The M matrix is first zeroed. If open loop control is desired, program control is returned to subroutine CONTRL. Otherwise, the elements are set within the M matrix as required for the feedback needed in the control technique defined in subroutine CONTRL.

Figure A3-16 is the VCLR for subroutine COMPM.

Zero the Matrix, M, Used to Relate Sensor Output to the Reference Input Signal	
T	F
.NOT. Open Loop Control	
Compute M Matrix Elements	Null

Figure A3-16 COMPM VCLR

3.2.2 Load Reference Signal (LOADR)

Subroutine LOADR is called from subroutine CONTRL to load the reference signal array, R. The reference array is first zeroed. The appropriate array elements are then set to the reference signal values valid for the current time.

Figure A3-17 is the VCLR for subroutine LOADR.

Zero the Reference Input Signal Array, R
Set R Array Elements:

Figure A3-17 LOADR VCLR

3.2.3 Compute Controller Phi Matrix (PHIC)

Subroutine PHIC is called from subroutine CONTRL to compute the state-variable formulation phi matrix for the controller. The phi matrix is first zeroed. The required elements of the phi matrix are then computed based on the controller equations placed in state-variable formulation form.

Figure A3-18 is the VCLR for subroutine PHIC.

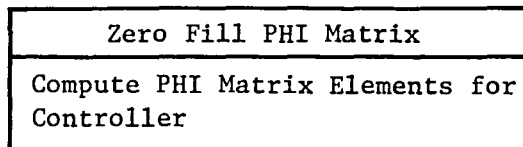


Figure A3-18 PHIC VCLR

3.2.4 Compute Controller Theta Matrix (THETAC)

Subroutine THETAC is called from subroutine CONTRL to compute the state-variable formulation theta matrix for the controller. The theta matrix is first zeroed. The required elements of the theta matrix are then computed based on the controller equations placed in state-variable formulation form.

Figure A3-19 is the VCLR for subroutine THETAC.

Zero Fill THETA Matrix
Compute THETA Matrix Elements for Controller

Figure A3-19 THETAC VCLR

3.2.5 Compute Controller C Matrix (CONTC)

Subroutine CONTC is called from subroutine CONTRL to compute the matrix, C, which transforms the controller state-variable array to the controller observable array. Under current implementation this matrix is always the identity matrix.

Figure A3-20 is the VCLR for subroutine CONTC.

Zero the Matrix, C, used to Transform the State Variable Array to the Observable Array
--

Compute C Matrix Elements

Figure A3-20 CONTC VCLR

3.2.6 Controller Process Noise (CNOISE)

Subroutine CNOISE is called from subroutine CONTRL to compute the state-variable noise for the controller. The noise array, W, is first zeroed. Subroutine GGNML of the IMSL math package is called to compute a normal random value from a Gaussian distribution with zero mean and variance of one. The controller state noise, W, is then computed as the user-specified standard deviation for controller state noise times the value obtained from subroutine GGNML.

Figure A3-21 is the VCLR for subroutine CNOISE.

Zero the Noise Array
CALL GGNML to Obtain a Sample from a Gaussian Distribution with Zero Mean and Variance of One
Compute Noise Array for the Control Array (User-Specified Standard Deviation Times Sample Value)

Figure A3-21 CNOISE VCLR

3.3 AMPLIFIER MODEL (AMP)

Subroutine AMP is called from the main routine JNTMOD to model the amplifier using the state-variable formulation.

Subroutine PHIA is called to compute the state-variable formulation phi matrix for the amplifier. Subroutine THETAA is called to compute the state-variable formulation theta matrix for the amplifier. Subroutine MATMPY is called to multiply the phi matrix and the current amplifier state-variable array, X, to produce the homogeneous state array, XHOM. Subroutine MATMPY is then called to multiply the theta matrix with the input control matrix, U, to produce the particular state array, XPAR. The state array without noise is the sum of XHOM and XPAP. Subroutine XNOISE is called to compute the amplifier process noise array, W. The process noise, W, plus XWON, produces the updated amplifier state variable array, X.

Subroutine COMPC is called to compute the amplifier state-to-observable transform matrix, C. Subroutine MATMPY is called to multiply the C matrix and the state array, X, to produce the observable array, Y, for the amplifier model.

Subroutine SENSOR is called to compute the sensor output without noise, ZWON; the sensor noise, V; and the sensor output including noise, Z.

Figure A3-22 is the VCLR for subroutine AMP.

T	PHI Matrix a Function of Time		F
CALL PHIA to Compute Amplifier PHI Matrix			Null
T	THETA Matrix a Function of Time		F
CALL THETAA to Compute Amplifier THETA Matrix			Null
CALL MATMPY to Multiply State Variable by PHI Matrix and Produce Homogeneous State			
CALL MATMPY to multiply Control Array by THETA Matrix and Produce Particular State			
CALL MATADD to Add Homogeneous to Particular State and Obtain State without Noise			
CALL XNOISE to Compute Noise Array for State Variable Array			
CALL MATADD to Add Noise to State without Noise and Produce Final State Variable			
T	C Matrix a Function of Time		F
CALL COMPC to Compute C Matrix Used to Transform State Variable to Observable Array			Null
CALL MATMPY to Multiply State Variable Array by C Matrix and Obtain Observable Array			
CALL SENSOR to Compute Sensor Noise and Sensor Output with and without Noise			

Figure A3-22 AMP VCLR

3.3.1 Compute Amplifier Phi Matrix (PHIA)

Subroutine PHIA is called from subroutine AMP to compute the state-variable formulation phi matrix for the amplifier. The phi matrix is first zeroed. The required elements of the phi matrix are then computed based on the amplifier equations placed in state-variable formulation form. Several types of amplifiers may be modeled. Input argument ITYPE is used to select the desired amplifier model.

Figure A3-23 is the VCLR for subroutine PHIA.

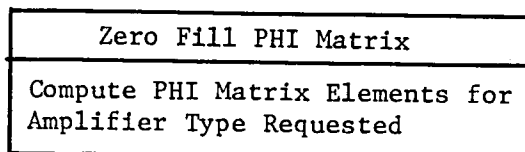


Figure A3-23 PHIA VCLR

3.3.2 Compute Amplifier Theta Matrix (THETAA)

Subroutine THETAA is called from subroutine AMP to compute the state-variable formulation theta matrix for the amplifier. The theta matrix is first zeroed. The required elements of the theta matrix are then computed based on the amplifier equations placed in state-variable formulation form. Several types of amplifiers may be modeled. Input argument ITYPE is used to select the desired amplifier model.

Figure A3-24 is the VCLR for subroutine THETAA.

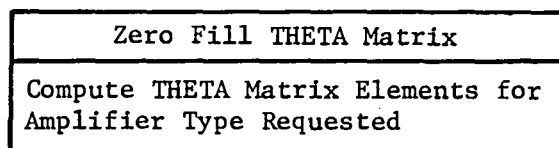


Figure A3-24 THETAA VCLR

3.4 MOTOR MODEL (MOTOR)

Subroutine MOTOR is called from the main routine JNTMOD to model the motor using the state-variable formulation.

Subroutine PHIM is called to compute the state-variable formulation phi matrix for the motor. Subroutine THETAM is called to compute the state-variable formulation theta matrix for the motor. Subroutine MATMPY is called to multiply the phi matrix and the current motor state variable array, X, to produce the homogeneous state array, XHOM. Subroutine MATMPY is then called to multiply the theta matrix with the input control matrix, U, to produce the particular state array, XPAR. The state array without noise is the sum of XHOM and XPAP. Subroutine XNOISE is called to compute the motor process noise array, W. The process noise, W, plus XWON, produces the updated motor state-variable array, X.

Subroutine COMPC is called to compute the motor state-to-observable transform matrix, C. Subroutine MATMPY is called to multiply the C matrix and the state array, X, to produce the observable array, Y, for the motor model.

Subroutine SENSOR is called to compute the sensor output without noise, ZWON; the sensor noise, V; and the sensor output including noise, Z.

Figure A3-25 is the VCLR for subroutine MOTOR.

T	PHI Matrix a Function of Time	F
CALL PHIM to Compute the Motor PHI Matrix		Null
T	THETA Matrix a Function of Time	F
CALL THETAM to Compute the Motor THETA Matrix		Null
CALL MATMPY to Multiply the State Variable Array, X by PHI and Produce X Homogeneous, XHOM		
CALL MATMPY to Multiply the Control Array, U, by THETA and Produce X Particular, XPAR		
CALL MATADD to Add XHOM to XPAR and Produce X without Noise, XWON		
CALL XNOISE to Compute the Noise Array, W, for X		
CALL MATADD to Add XWON to W and Produce the Final State Variable, X		
T	C Matrix a Function of Time	F
CALL COMPC to Compute the Matrix, C, Used to Transform X to the Observable Array, Y		Null
CALL MATMPY to Multiply X by C and Produce the Observable Array, Y		
CALL SENSOR to Compute the Sensor Output without Noise, ZWON; The Sensor Noise, V; and the Sensor Output Including Noise, Z		

Figure A3-25 MOTOR VCLR

3.4.1 Compute Motor Phi Matrix (PHIM)

Subroutine PHIM is called from subroutine MOTOR to compute the state-variable formulation phi matrix for the motor. The phi matrix is first zeroed. The required elements of the phi matrix are then computed based on the motor equations placed in state-variable formulation form. Several types of motors may be modeled. Input argument ITYPE is used to select the desired motor model.

Figure A3-26 is the VCLR for subroutine PHIM.

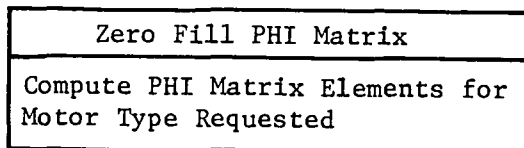


Figure A3-26 PHIM VCLR

3.4.2 Compute Motor Theta Matrix (THETAM)

Subroutine THETAM is called from subroutine MOTOR to compute the state-variable formulation theta matrix for the motor. The theta matrix is first zeroed. The required elements of the theta matrix are then computed based on the motor equations placed in state-variable formulation form. Several types of motors may be modeled. Input argument ITYPE is used to select the desired motor model.

Figure A3-27 is the VCLR for subroutine THETAM.

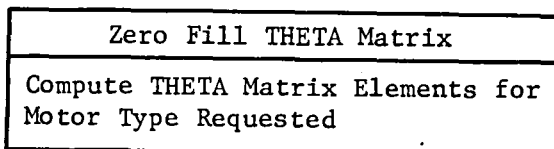


Figure A3-27 THETAM VCLR

3.5 POWER TRAIN MODEL (PWRTRN)

Subroutine PWRTRN is called from the main routine JNTMOD to model the power train using the state-variable formulation.

Subroutine PHIP is called to compute the state-variable formulation phi matrix for the power train. Subroutine THETAP is called to compute the state-variable formulation theta matrix for the power train. Subroutine MATMPY is called to multiply the phi matrix and the current power train state-variable array, X, to produce the homogeneous state array, XHOM. Subroutine MATMPY is then called to multiply the theta matrix with the input control matrix, U, to produce the particular state array, XPAR. The state array without noise is the sum of XHOM and XPAP. Subroutine XNOISE is called to compute the power train process noise array, W. The process noise, W, plus XWON, produces the updated power train state-variable array, X.

Subroutine COMPC is called to compute the power train state-to-observable transform matrix, C. Subroutine MATMPY is called to multiply the C matrix and the state array, X, to produce the observable array, Y, for the power train model.

Subroutine SENSOR is called to compute the sensor output without noise, ZWON; the sensor noise, V; and the sensor output including noise, Z.

Figure A3-28 is the VCLR for subroutine PWRTRN.

T	PHI Matrix a Function of Time	F
CALL PHIP to Compute the Power Train PHI Matrix		Null
T	THETA Matrix a Function of Time	F
CALL THETAP to Compute the Power Train THETA Matrix		Null
CALL MATMPY to Multiply the State Variable Array, X by PHI and Produce X Homogeneous, XHOM		
CALL MATMPY to Multiply the Control Array, U, by THETA and Produce X Particular, XPAR		
CALL MATADD to Add XHOM to XPAR and Produce X without Noise, XWON		
CALL XNOISE to Compute the Noise Array, W, for X		
CALL MATADD to Add XWON to W and Produce the Final State Variable Array, X		
T	C Matrix a Function of Time	F
CALL COMPC to Compute the Matrix, C, Used to Transform X to the Observable Array, Y		Null
CALL MATMPY to Multiply X by C and Produce the Observable Array, Y		
CALL SENSOR to Compute the Sensor Output without Noise, ZWON; the Sensor Noise, V; and the Sensor Output Including Noise, Z		

Figure A3-28 PWRTRN VCLR

3.5.1 Compute Power Train Phi Matrix (PHIP)

Subroutine PHIP is called from subroutine PWRTRN to compute the state-variable formulation phi matrix for the power train. The phi matrix is first zeroed. The required elements of the phi matrix are then computed based on the power train equations placed in state-variable formulation form. Several types of power trains may be modeled. Input argument ITYPE is used to select the desired power train model.

Figure A3-29 is the VCLR for subroutine PHIP.

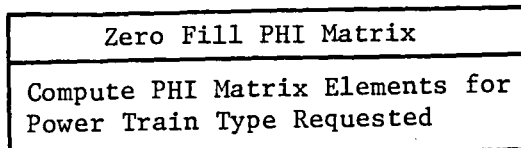


Figure A3-29 PHIP VCLR

3.5.2 Compute Power Train Theta Matrix (THETAP)

Subroutine THETAP is called from subroutine PWRTRN to compute the state-variable formulation theta matrix for the power train. The theta matrix is first zeroed. The required elements of the theta matrix are then computed based on the power train equations placed in state variable formulation form. Several types of power trains may be modeled. Input argument ITYPE is used to select the desired power train model.

Figure A3-30 is the VCLR for subroutine THETAP.

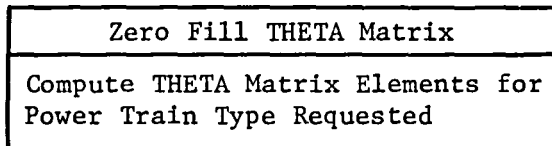


Figure A3-30 THETAP VCLR

3.6 LOAD MODEL (LOAD)

Subroutine LOAD is called from the main routine JNTMOD to model the load using the state-variable formulation.

Subroutine PHIL is called to compute the state-variable formulation phi matrix for the load. Subroutine THETAP is called to compute the state-variable formulation theta matrix for the load. Subroutine MATMPY is called to multiply the phi matrix and the current load state variable array, X, to produce the homogeneous state array, XHOM. Subroutine MATMPY is then called to multiply the theta matrix with the input control matrix, U, to produce the particular state array, XPAR. The state array without noise is the sum of XHOM and XPAP. Subroutine XNOISE is called to compute the load process noise array, W. The process noise, W, plus XWON, produces the updated load state-variable array, X.

Subroutine COMPC is called to compute the load state-to-observable transform matrix, C. Subroutine MATMPY is called to multiply the C matrix and the state array, X, to produce the observable array, Y, for the load model.

Subroutine SENSOR is called to compute the sensor output without noise, ZWON, the sensor noise, V, and the sensor output including noise, Z.

Figure A3-31 is the VCLR for subroutine LOAD.

T	PHI Matrix a Function of Time	F
CALL PHIL to Compute the Load PHI Matrix		Null
T	THETA Matrix a Function of Time	F
CALL THETAL to Compute the Load THETA Matrix		Null
CALL MATMPY to Multiply the State Variable, Array, X, by PHI and Produce X Homogeneous, XHOM		
CALL MATMPY to Multiply the Control Array, U, by THETA and Produce X Particular, XPAR		
CALL MATADD to Add XHOM to XPAR and Produce X without Noise, XWON		
CALL XNOISE to Compute the Noise Array, W, for X		
CALL MATADD to Add XWON to W and Produce the Final State Variable Array, X		
T	C Matrix a Function of Time	F
CALL COMPC to Compute the Matrix, C, Used to Transform X to the Observable Array, Y		Null
CALL MATMPY to Multiply X by C and Produce the Observable Array, Y		
CALL SENSOR to Compute the Sensor Output without Noise, ZWON; the Sensor Noise, V; and the Sensor Output Including Noise, Z		

Figure A3-31 LOAD VCLR

3.6.1 Compute Load Phi Matrix (PHIL)

Subroutine PHIL is called from subroutine LOAD to compute the state-variable formulation phi matrix for the load. The phi matrix is first zeroed. The required elements of the phi matrix are then computed based on the load equations placed in state-variable formulation form. Several types of loads may be modeled. Input argument ITYPE is used to select the desired load model.

Figure A3-32 is the VCLR for subroutine PHIL.

Zero Fill PHI Matrix
Compute PHI Matrix Elements for Load Type Requested

Figure A3-32 PHIL VCLR

3.6.2 Compute Load Theta Matrix (THETAL)

Subroutine THETAL is called from subroutine LOAD to compute the state-variable formulation theta matrix for the load. The theta matrix is first zeroed. The required elements of the theta matrix are then computed based on the load equations placed in state-variable formulation form. Several types of loads may be modeled. Input argument ITYPE is used to select the desired load model.

Figure A3-33 is the VCLR for subroutine THETAL.

Zero Fill THETA Matrix
Compute THETA Matrix Elements for Load Type Requested

Figure A3-33 THETAL VCLR

3.7 KALMAN FILTER

This section describes all routines related to the Kalman filter implemented within the JNTMOD program.

3.7.1 Kalman Filter Initialization (KFINIT)

Subroutine KFINIT is called from the main routine JNTMOD to perform initialization of the matrices and arrays used in the Kalman filter.

Subroutine BLDPHI is called to compute the system phi matrix element for the simulation start time. Subroutine MATTRN is called to compute the transpose of the system phi matrix. Subroutine BUILDH is called to compute the system sensor transform matrix at the simulation start time. Subroutine MATTRN is called to compute the transpose of the system sensor matrix. Subroutine BLDTHT is called to compute the elements of the system theta matrix at the simulation start time.

The P, Q, and R inverse matrices are zeroed. The diagonal elements of the P matrix are set to 100.0. The diagonal elements of the Q matrix are set to the process noise variance values. The diagonal elements of the R inverse matrix are set to the reciprocal of the sensor variance values.

The XHAT array elements are initialized to zero. The H transpose-R inverse-H matrix is computed as is the H transpose-R inverse matrix.

Figure A3-34 is the VCLR for subroutine KFINIT.

CALL BLDPHI to Compute Initial System PHI Matrix, Φ
CALL MATTRN to Compute Transpose of Initial System PHI Matrix, Φ^T
CALL BUILDH to Compute Initial System Sensor Matrix, H
CALL MATTRN to Compute Transpose of Initial System Sensor Matrix, H^T
CALL BLDTHT to Compute Initial System THETA Matrix, Θ
Initialize P Matrix with 100.0 along Diagonal
Set Diagonal Elements of Q Matrix to System Noise Variance Values
Set Diagonal Elements of R^{-1} Matrix to the Inverse of the System Sensor Noise Variance Values
Initialize \hat{x} Values to Zero
Compute Initial Value of $H^T R^{-1} H$
Compute Initial Value of $H^T R^{-1}$

Figure A3-34 KFINIT VCLR

3.7.2 Build System Sensor Array (BLDZAL)

Subroutine BLDZAL loads the system sensor output array ZALL. The elements of the sensor output arrays from the amplifier, motor, power train, and load models are combined to form the ZALL array.

Figure A3-35 is the VCLR for subroutine BLDZAL.

Zero System Sensor Array
Build System Sensor Array Using Sensor Output Arrays from Amp, Motor, Power Train, and Load

Figure A3-35 BLDZAL VCLR

3.7.3 Build System Phi Matrix (BLDPHI)

Subroutine BLDPHI loads the system phi matrix needed for the Kalman filter used in conjunction with the state-variable formulation. The system phi matrix is constructed using matrices from each of the joint model blocks as follows:

$$\Phi = \begin{bmatrix} \phi_C & 0 & 0 & 0 & 0 \\ 0 & \phi_A & & & 0 \\ 0 & & \phi_M & & \\ 0 & & & \phi_P & \\ 0 & 0 & & & \phi_L \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ \theta_A & & & 0 \\ & \theta_M & & \\ & & \theta_P & \\ 0 & & & \theta_L \end{bmatrix} \begin{bmatrix} D_A \\ D_M \\ D_P \\ D_L \end{bmatrix} \begin{bmatrix} C_C & 0 & 0 & 0 & 0 \\ 0 & C_A & & & 0 \\ 0 & & C_M & & \\ 0 & & & C_P & \\ 0 & 0 & & & C_L \end{bmatrix}$$

where the subscripts are defined as

C = Controller;

A = Amplifier;

M = Motor;

P = Power train;

L = Load.

Figure A3-36 is the VCLR for subroutine BLDPHI.

Zero All Output Matrices
Build Script Phi Matrix Using Phi Matrices from Controller, Amp, Motor, Power Train, and Load as Block Diagonals
Build Script Theta Matrix Using Theta Matrices from Controller, Amp, Motor, Power Train, and Load as Block Diagonals
Build System D Matrix
Build System C Matrix
CALL MATMPY to Compute Product of System D Matrix and System C Matrix
CALL MATMPY to Multiply Result by Script Theta Matrix
CALL MATADD to Add Result to Script Phi Matrix, Producing System Phi Matrix

Figure A3-36 BLDPHI VCLR

3.7.4 Build System Theta Matrix (BLDTHT)

Subroutine BLDTHHT loads the system theta matrix needed for the Kalman filter used in conjunction with the state-variable formulation. The system theta matrix is constructed using matrices from the joint model controller block as follows:

$$\Theta = \begin{bmatrix} \theta_C & | & \theta_C^M \\ \hline 0 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix}$$

where the subscripts are defined as

C = Controller.

Figure A3-37 is the VCLR for subroutine BLDTHHT.

Zero Output Matrix
CALL MATMPY to Multiply Controller Theta Matrix and M Matrix
Load Elements of Result and Controller Theta Matrix into System Theta Matrix

Figure A3-37 BLDTHHT VCLR

3.7.5 Build System Sensor Transform Matrix (BUILDH)

Subroutine BUILDH loads the system sensor transform matrix needed for the Kalman filter used in conjunction with the state-variable formulation. The system sensor transform matrix is constructed using matrices from each of the joint model blocks as follows:

$$H = \begin{bmatrix} 0 & H_A & & 0 \\ 0 & & H_M & \\ 0 & & & H_P \\ 0 & 0 & & H_L \end{bmatrix}$$

where the subscripts are defined as

A = Amplifier;

M = Motor;

P = Power train;

L = Load.

Figure A3-38 is the VCLR for subroutine BUILDH.

Zero Script H Matrix (System Sensor Matrix)
Build Script H Matrix Using H Matrices from Amp, Motor, Power Train, and Load as Block Diagonals

Figure A3-38 BUILDH VCLR

3.7.6 Kalman Filter Computations (KALMAN)

Subroutine KALMAN is called from the main routine JNTMOD to perform the Kalman filter calculations. Subroutine BUILDG is called to compute the G matrix. All other matrices and arrays are available for use. The matrix equations used in subroutine KALMAN are as follows:

$$\hat{z}(k+1) = H(k+1) \phi(k+1, k) \hat{x}(k/k)$$

$$\hat{x}(k+1/k+1) = \phi(k+1, k) \hat{x}(k/k) + \theta(k+1, k) \alpha(k) + G(k+1) \{z(k+1) - \hat{z}(k+1)\}$$

where

$\hat{z}(k+1)$ is the best estimate of the system sensor output

$\hat{x}(k+1/k+1)$ is the best estimate of the system state

Subroutine PRTKF is called if printed output of the Kalman filter results is desired.

Figure A3-39 is the VCLR for subroutine KALMAN.

CALL BUILDG to Compute Kalman Gain Matrix G	
CALL MATMPY to Compute $\phi \hat{x}$	
CALL MATMPY to Compute $\hat{z} = H \phi \hat{x}$	
CALL MATSUB to Compute $z - \hat{z}$	
CALL MATMPY to Compute $G(z - \hat{z})$	
CALL MATMPY to Compute $\theta \alpha$	
CALL MATADD to Compute $\theta \alpha + G(z - \hat{z})$	
CALL MATMPY to Compute $\phi \hat{x}$	
CALL MATADD to Compute $\phi \hat{x} + \theta \alpha + G(z - \hat{z}) = \hat{x}$	
T	Print File Requested F
CALL PRTKF to Print Kalman Filter Results	Null

Figure A3-39 KALMAN VCLR

3.7.6.1 Build G Matrix (BUILDG) - Subroutine BUILDG is called from subroutine KALMAN to build the G matrix needed in the Kalman filter used in conjunction with the state-variable formulation. The following equations define the G matrix.

$$P(k+1/k) = \Phi(k+1,k)P(k/k)\Phi^T(k+1,k) + Q(k)$$

$$P(k+1/k+1) = \{P(k+1/k)^{-1} + H^T R^{-1} H\}^{-1}$$

$$G(k+1) = P(k+1/k+1)H^T(k+1)R^{-1}(k+1)$$

Figure A3-40 is the VCLR for subroutine BUILDG.

CALL MATTRN to Compute Φ^T
CALL MATTRN to Compute H^T
CALL MATMPY to Compute $P \Phi^T$
CALL MATMPY to Compute $\Phi P \Phi^T$
CALL MATADD to Compute $P = \Phi P \Phi^T + Q$
CALL LGINF to Compute P^{-1}
CALL MATMPY to Compute $P^{-1} H$
CALL MATMPY to Compute $H^T P^{-1} H$
CALL MATADD to Compute $P^{-1} + H^T P^{-1} H$
CALL LGINF to Compute $(P^{-1} + H^T R^{-1} H)^{-1} = P$
CALL MATMPY to Compute $H^T P^{-1}$
CALL MATMPY to Compute $P H^T R^{-1} = G$

Figure A3-40 BUILDG VCLR

3.7.6.2 Print Kalman Filter Results (PRTKF) - Subroutine PRTKF is called from subroutine KALMAN to print the Kalman filter results. At each call to subroutine PRTKF, the following parameters are printed:

- 1) Sensor output arrays--Z, ZHAT, and Z-ZHAT;
- 2) State-variable arrays--XHAT and SIGMAX.

Figure A3-41 is the VCLR for subroutine PRTKF.

Start New Page and Print Page Header with Current Time
Print Sensor Output
Print State Variable Output of Kalman Filter

Figure A3-41 PRTKF VCLR

3.8 PLOT FILE ROUTINES

This subsection describes the routines required to produce the plot file. The plot file contains all of the data generated in the joint model and Kalman filter routines. The contents of the plot file can be displayed as parameter versus parameter plots using the general plotting routine GENPLT.

3.8.1 Generate Symbol Record (PLTSET)

Subroutine PLTSET is called from the main routine JNTMOD to generate the symbol record for the plot file. The symbol record contains character symbols in the same location within the symbol record that is occupied by the corresponding data in the data records. The symbols in the symbol record allow the corresponding data to be easily located by routines that read the plot file.

Symbols for all possible plot parameters are stored in DATA statements within subroutine PLTSET. The symbol record array elements are loaded from the data in these DATA statements using the system dimension parameters for the joint system being modeled. After the symbol record is filled, the plot file is opened as an unformatted direct access file. A dummy header record is written followed by the symbol record.

Figure A3-42 is the VCLR for subroutine PLTSET.

Set Logical Unit Numbers	
Load Reference Parameter Mnemonics into Symbol Record	
Load Control Parameter Mnemonics into Symbol Record	
Load Amplifier Parameter Mnemonics into Symbol Record	
Load Motor Parameter Mnemonics into Symbol Record	
Load Power Train Parameter Mnemonics into Symbol Record	
Load Load Parameter Mnemonics into Symbol Record	
T	F
Kalman Filter Being Used	
Load Kalman Filter Parameter Mnemonics into Symbol Record	Null
Open Plot File	
Write Dummy Header Record to File	
Write Symbol Record to File	

Figure A3-42 PLTSET VCLR

3.8.2 Load Plot File Record (LDVALU)

Subroutine LDVALU is called from the main routine JNTMOD to load values into the plot file record. Subroutine LDVALU loads the values in input array A into the next available locations within the plot record. The values of array A are checked against the current maximum and minimum values corresponding to the array parameters. The maximum and minimum values are updated if necessary.

Figure A3-43 is the VCLR for subroutine LDVALU.

Load Plot Output Array with Values of Current Time
Update Max/Min of Values Stored for each Parameter

Figure A3-43 LDVALU VCLR

3.8.3 Write Header Record (WRTHDR)

Subroutine WRTHDR is called from the main routine JNTMOD after the simulation time loop is finished. Subroutine WRTHDR writes the header record to the plot file. The header record contains the maximum and minimum values for each parameter in the plot file data records.

Figure A3-44 is the VCLR for subroutine WRTHDR.

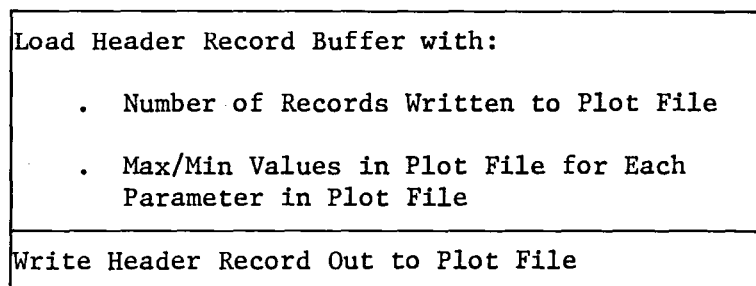


Figure A3-44 WRTHDR VCLR

3.9 MATRIX MATH ROUTINES

This subsection contains the descriptions of the matrix math routines used throughout the JNTMOD program. The routines documented in this section are:

- 1) MATADD - Matrix addition;
- 2) MATMPY - Matrix multiplication;
- 3) MATTRN - Matrix transpose;
- 4) MATSUB - Matrix subtraction.

All matrix routines use variably dimensioned arrays that allow matrix, vector, and scalar operations.

3.9.1 Matrix Addition (MATADD)

Subroutine MATADD adds input matrix A to input matrix B to produce output matrix C. All arrays are variably dimensioned, which allows matrix, vector, or scalar operation.

Figure A3-45 is the VCLR for subroutine MATADD.

Add the Two Input Matrices (or Vectors)

Figure A3-45 MATADD VCLR

3.9.2 Matrix Multiplication (MATMPY)

Subroutine MATMPY multiplies input matrix A and input matrix B to produce output matrix C. All arrays are variably dimensioned, which allows matrix, vector, or scalar operation.

Figure A3-46 is the VCLR for subroutine MATMPY.

Zero the Output Array
Multiply the Two Input Matrices (or Vectors)

Figure A3-46 MATMPY VCLR

3.9.3 Matrix Transpose (MATTRN)

Subroutine MATTRN loads the output matrix B with the transpose of input matrix A.

Figure A3-47 is the VCLR for subroutine MATTRN.

Compute the Transpose of the Input Matrix

Figure A3-47 MATTRN VCLR

3.9.4 Matrix Subtraction (MATSUB)

Subroutine MATSUB subtracts input matrix B from input matrix A to produce output matrix C. All arrays are variably dimensioned, which allows matrix, vector, or scalar operation.

Figure A3-48 is the VCLR for subroutine MATSUB.

Subtract the Second Input Matrix (or Vector) from the First Input Matrix (or Vector)

Figure A3-48 MATSUB VCLR

3.10 IMSL MATH PACKAGE ROUTINES

This subsection briefly describes the two math routines used from the IMSL math package.

- 1) CALL GGNML - Compute normal random value from a Gaussian distribution with zero mean and variance of one;
- 2) CALL LGINF - Compute matrix inverse or pseudo-inverse.

1. Report No. NASA CR-165976		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle EVALUATION OF AUTOMATED DECISIONMAKING METHODOLOGIES AND DEVELOPMENT OF AN INTEGRATED ROBOTIC SYSTEM SIMULATION-APPENDIX A, SOFTWARE DOCUMENTATION				5. Report Date September 1982	
				6. Performing Organization Code	
7. Author(s) J. W. Lowrie, Dr. A. Fermelia, D. C. Haley, K. D. Gremban, J. Van Baalen, R. W. Walsh				8. Performing Organization Report No. MCR-82-581 Vol. II Part 1	
				10. Work Unit No.	
9. Performing Organization Name and Address Martin Marietta Aerospace Denver Division P.O. Box 179 Denver, CO 80201				11. Contract or Grant No. NAS1-16759	
				13. Type of Report and Period Covered Contractor Report	
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546				14. Sponsoring Agency Code	
15. Supplementary Notes Langley Technical Monitor: Jack Pennington Final Report					
16. Abstract This report contains Appendix A to NASA Contractor Report NASA CR-165975. Appendix A contains documentation of the preliminary software developed as a framework for a generalized integrated robotic system simulation. The program structure is composed of three major functions controlled by a program executive. The three major functions are: system definition, analysis tools, and post processing. The system definition function handles user input of system parameters and definition of the manipulator configuration. The analysis tools function handles the computational requirements of the program. The post processing function allows for more detailed study of the results of analysis tool function executions. Also documented is the manipulator joint model software to be used as the basis of the manipulator simulation which will be part of the analysis tools capability.					
17. Key Words (Suggested by Author(s)) robotic simulation, software			18. Distribution Statement Unclassified--Unlimited		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 167	22. Price		

End of Document